175 PTAS
97
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
AUTUMN HERBICIDES
Suspect
Hoechst
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IX-Fascículo 97

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-181-X (tomo 9)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 048512
Impreso en España-Printed in Spain-Diciembre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.



Bajo sospecha

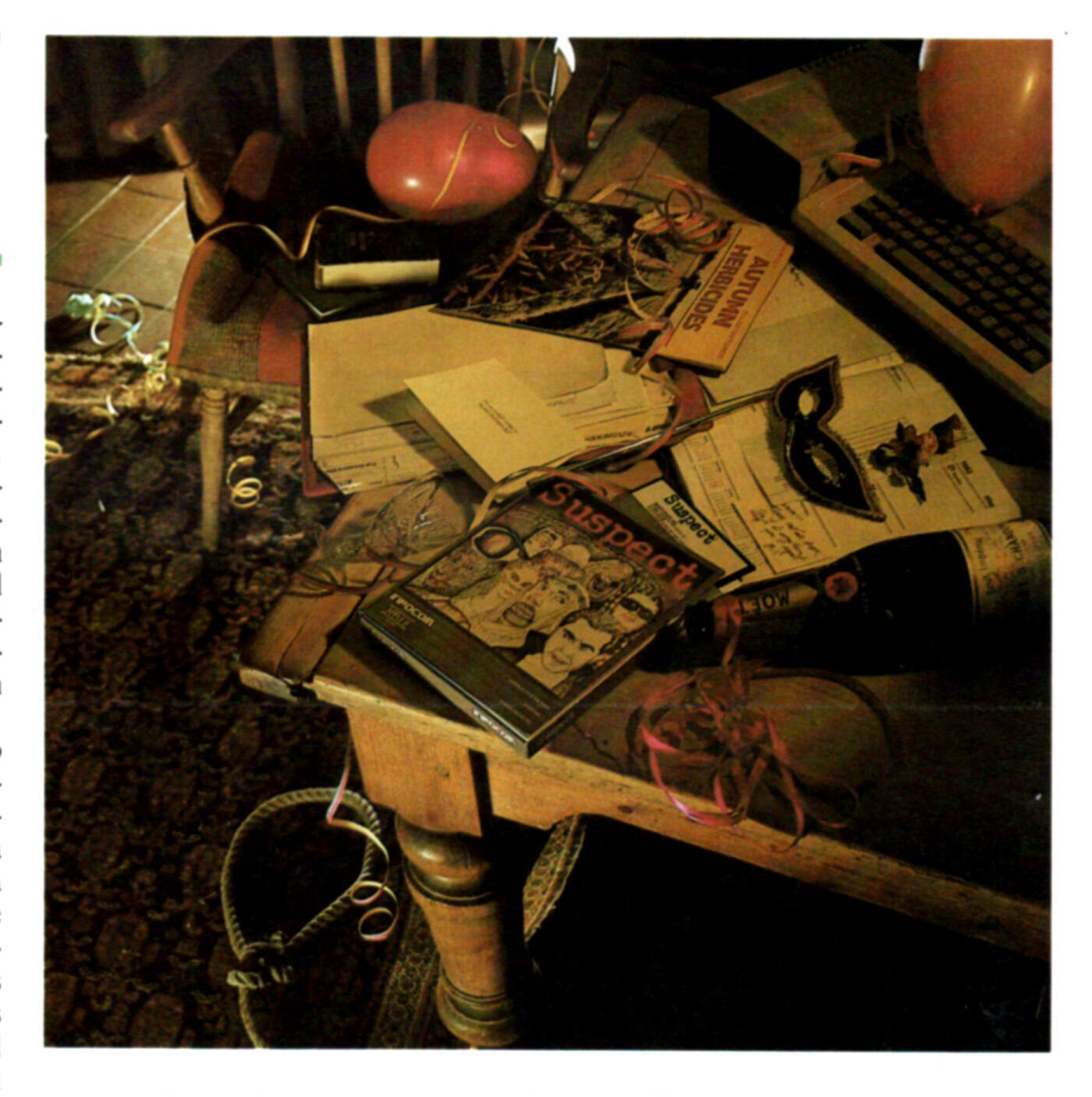
La programación de juegos de aventuras alcanza cotas de notable refinamiento al incluir personajes interactivos

El software que parece dotar de una «personalidad» al ordenador, mediante el análisis de las entradas del usuario en su lenguaje natural y la producción de respuestas aparentemente llenas de significado, logra siempre generar interés y emoción. Cualesquiera que sean las pautas psicológicas implicadas, programas como ELIZA permiten al usuario un nivel de participación que no guarda relación en absoluto con su relativa simplicidad y, desde el punto de vista del programador, existen pocos desafíos más satisfactorios que el de programar un ordenador para que «escuche» y «responda» como un compañero humano.

Ya hemos desarrollado un programa tipo ELIZA muy sencillo (*Consulta particular*) y, además analizado los principios del proceso del lenguaje natural a lo largo de nuestra serie dedicada a la inteligencia artificial. Ahora examinaremos otra aplicación de estos principios: la programación de «personajes interactivos» en los juegos de aventuras. Analizaremos el papel que pueden jugar tales personajes en estos juegos y luego programaremos nuestro propio manipulador de personajes, el cual se podrá ejecutar como módulo individual, o bien adaptar para ejecutarlo junto con nuestros programas *El bosque encantado* y *Digitaya*.

El nivel de sofisticación del software de aventuras ha ido en aumento desde que Crowther y Woods desarrollaran *Colossal cave* (Caverna colosal) en FORTRAN y en un ordenador de unidad principal, utilizando aproximadamente 170 Kbytes de código. En la actualidad, programas como *The hitchhiker's guide to the galaxy* (Guía de la galaxia para el autostopista) no sólo aceptan entradas complejas sino que también cuentan con personajes que actúan «espontáneamente» y a los que el jugador puede dirigirles la palabra (e incluso, en algunos casos, impartirles instrucciones).

La inclusión de personajes en la *ficción interactiva* (tal como se está empezando a llamar al software de aventuras) les ha permitido a los programadores alejarse del ambiente tradicional de aventuras, que se centraba casi exclusivamente en el proceso de explorar distintos escenarios, recoger objetos y resolver el ocasional enigma. Los personajes interactivos introducen nuevas posibilidades; en primer lugar, la presentación de situaciones que de hecho el jugador no puede resolver sin la ayuda de otro personaje controlado por el ordenador; en segundo lugar, y aun más importante, la presentación de un juego que puede resultar radicalmente diferente cada vez que se juega con él.



El dedo acusador lo señala a usted
Una reina de cuento de hadas, un hombre lobo, un vampiro y muchos otros personajes lo esperan a usted en *Suspect* (Sospechoso), un juego de aventuras de Infocom que incorpora personajes interactivos y vívidas descripciones de escenarios. Usted, invitado a un baile de disfraces de la alta sociedad, tropieza con la escena descrita arriba y descubre, con espanto, que su anfitriona ha sido estrangulada con un lazo que alguien le ha quitado de su disfraz de vaquero. Para salvar su reputación, usted habrá de interactuar de forma a la vez intensa y sutil con el resto de los asistentes al baile

Antes de seguir adelante es necesario que definamos exactamente qué es un personaje de estas características. Un personaje interactivo ha de poseer al menos uno de los siguientes atributos:

- Debe poder desplazarse de un escenario a otro.
- Debe poder recoger y abandonar objetos sin que el jugador tenga que instruirlo en ese sentido.
- Debe poder ser aludido por el jugador y responder de forma significativa (aunque sólo replique «No te comprendo»).
- Debe estar capacitado para dirigirse al jugador sin que éste así se lo indique.
- Debe ser consciente de su entorno y sensible al mismo.

A nivel ideal, el personaje interactivo ha de combinar todos estos atributos. Quizá el mejor ejemplo de los inicios de la programación de personajes interactivos sea *El Hobbit*, una aventura de texto y gráficos de Melbourne House. Este juego incorporaba numerosos personajes, incluyendo, de forma destacable, a Thorin el Enano y Gandalf el Mago. El jugador podía dirigirse a estos personajes mediante la instrucción SAY TO (decir a), como en SAY TO THORIN «GO NORTH» (decir a Thorin «avanzar norte»). El programa empleaba entonces una rutina activada por números aleatorios que decidía si el personaje había de obedecer o no. No obstante, los

personajes de *El Hobbit* eran, para los estándares actuales, comparativamente primitivos: al impartirles una orden con frecuencia el resultado era: THORIN SAYS NO (Thorin dice no) o una respuesta similar. Además, Gandalf y sus compañeros no hablaban particularmente bien, o de hecho no hablaban en absoluto. Thorin, por ejemplo, parece estar limitado a decir HURRY UP (date prisa) si el jugador se demora demasiado tiempo entre las entradas, y tiene la costumbre de «sentarse a pensar en los peces de colores» con monótona frecuencia.

Quizá lo más significativo sea el hecho de que los personajes de *El Hobbit* no poseen ninguna historia personal, es decir que si uno efectúa alguna acción determinada que pueda incidir en su relación con ellos, los personajes posteriormente no la «recordarán». Entre la mayoría de las primeras aventuras es típico que cada personaje esté programado para comportarse de un cierto modo a lo largo del juego y, en gran medida, independientemente de las circunstancias. De modo tal que Thorin, por ejemplo, podría «sentarse a pensar en los peces de colores» en el momento en que se le aproxima un dragón.

En *Valhalla*, programa que obtuvo en Gran Bretaña el premio Microcomputer Game of the Year en su edición de 1984, se efectuó un primer intento por hacer a los personajes dueños de una «historia»; al otorgársele la distinción se premió tanto la incorporación de esta facilidad como su visualización animada. En este juego, cuya acción tiene lugar entre dioses y héroes de la mitología escandinava, uno se encuentra con que los personajes «buenos» siempre se muestran dispuestos a obedecer las instrucciones del jugador (y viceversa) si el objetivo principal de éste es el de acabar con todos los personajes malvados.



La mansión del horror
Suspect se desarrolla en la exclusiva mansión Ashcroft, reflejada en la ilustración. Algunas de las habitaciones más grandes aparecen en el juego como varios escenarios diferentes; por ejemplo, el Salón de Baile junto a la Orquesta, el Salón de Baile junto al Hogar y el Recibidor Grande del Norte. El mapa ilustra la posición de algunos de los protagonistas inmediatamente después del asesinato de Verónica Ashcroft, perpetrado en el despacho

Clave:
1 Arlequín, mirando la televisión
2 Reina de las Hadas (muerta)
3 Gorila (el mayordomo, Smythe)
4 Jeque, esposo de la Reina de las Hadas
5 Muchacha del Harén, bailando
6 Astronauta, dirigiéndose al oeste
7 Barman
8 Vampiro, saliendo del salón de baile
9 Jugador, disfrazado de vaquero

Ello ofreció, entre otras cosas, la interesante posibilidad de personajes susceptibles de someterse a «lavados de cerebro». Por ejemplo, el dios escandinavo Loki inicialmente es un ente maligno, pero uno puede congraciarse con él provocando peleas con, supongamos, Thor (un dios virtuoso) y ordenándole después a Loki que realice buenas acciones, como enzarzarse en peleas con quienes en otro tiempo eran sus siniestros compañeros. Lamentablemente, a pesar de la complejidad del aspecto «histórico» de los personajes de *Valhalla*, los otros atributos de las personalidades participantes se limitan en su mayor parte, a comer, luchar y beber.

Veamos ahora una aventura interactiva muchísimo más avanzada: *Suspect* (Sospechoso), de Infocom (productora asimismo de *The hitchhiker's...)*. *Suspect* constituye un ejemplo perfecto del juego en el cual los personajes tienen suma importancia; de hecho, el éxito o el fracaso de éste depende exclusivamente de observar, interrogar, examinar y finalmente acusar a otras «personas» que se hallan bajo el control del ordenador. En consecuencia, y debido a los elementos aleatorios implicados, el juego se puede practicar de numerosas formas.



«Suspect»

La trama de *Suspect* se desarrolla en una mansión campestre (véase el mapa) e incluye al menos 12 protagonistas. La acción transcurre entre las nueve en punto de la noche y las primeras horas del día siguiente, tiempo durante el cual los Ashcrofts (millonarios norteamericanos de la alta sociedad) celebran su baile anual de disfraces. En el transcurso del juego usted descubre (si se molesta en echar una mirada por la casa) que Verónica Ashcroft ha sido estrangulada con el lazo que formaba parte del disfraz de vaquero que usted llevaba. Las sospechas, por consiguiente, recaen ineludiblemente sobre sus hombros y, a menos que descubra al verdadero culpable, pronto se verá entre rejas.

A lo largo del juego, los personajes deambulan por la casa, hablando entre ellos y viviendo su propia existencia. El programa le informa puntualmente a usted acerca de los entornos inmediatos de cada uno. Nuestro mapa muestra las posiciones que ocupan algunos de los personajes en un momento determinado del juego en el cual usted, de pie en el salón de baile junto a la orquesta, recibe la siguiente descripción en pantalla:

Salón de baile, junto a la orquesta
Éste es el extremo norte del salón de baile. Una superficie elevada constituye una plataforma para la orquesta, y hay una red de circuitos estéreo para usar cuando no hay orquesta. En otras partes de este gran salón de baile hay una multitud de otros invitados a la fiesta vestidos con toda clase de extravagantes indumentarias. En la pista de baile se hallan algunos de los bailarines de mayor edad. La orquesta está tocando el «*Tennessee waltz*». En la periferia de la habitación se observan pequeños grupos que charlan acerca de toda clase de temas, desde política hasta los escándalos locales. Michael se halla junto a la entrada norte. Alicia está en la pista de baile. Ostmann se encuentra junto a la entrada sur. Junto al hogar está el Astronauta. Johnson está en el bar. La orquesta está aquí, haciendo su juego. Ahora el Astronauta se halla junto a la entrada sur.

Observará que en la descripción se menciona dos veces al Astronauta: primero se halla junto al hogar y luego junto a la entrada sur, de modo que usted puede deducir que probablemente en ese momento esté marchándose del salón de baile. Las posiciones de otros personajes, como Smythe, el mayordomo, por supuesto no se mencionan; éstos se hallan en otras habitaciones y usted habrá de buscarlos para ser notificado de su presencia.

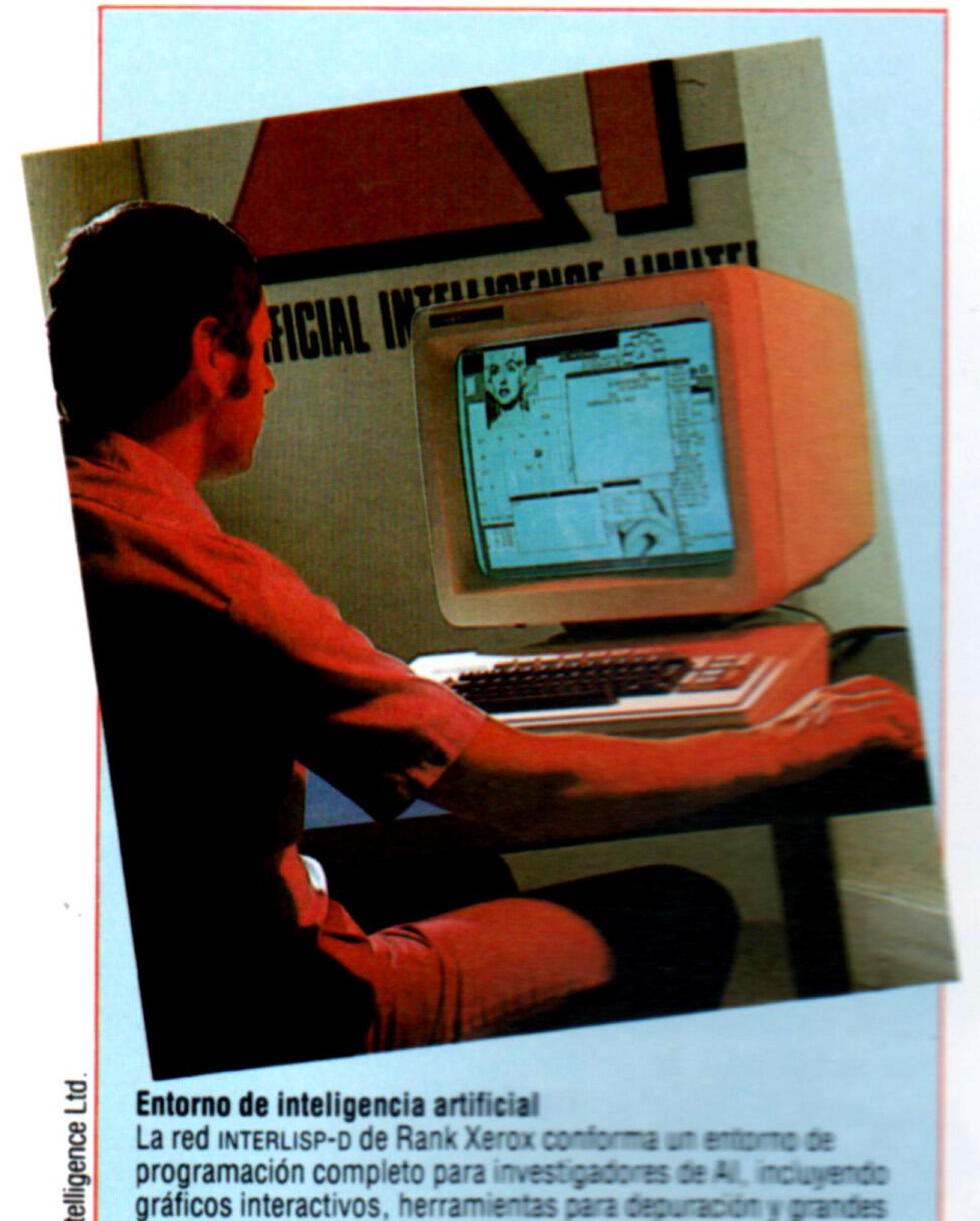
Jugar a un juego como *Suspect* constituye una experiencia apasionante, y la presencia de personajes interactivos le añade muchas dimensiones nuevas a un área de la programación de juegos cuya popularidad ya es notable. En el próximo capítulo examinaremos otros ejemplos de la interacción de personajes en el juego y daremos los primeros pasos para escribir nuestras propias rutinas y conseguir efectos similares.

El más adecuado

Veamos cuáles son los lenguajes de programación más apropiados para las aplicaciones de inteligencia artificial (AI)

La mayor parte de los investigadores en el campo de la inteligencia artificial (AI) emplean LISP o PROLOG. Por este motivo, estos dos lenguajes de programación han llegado a conocerse como *lenguajes de AI*. Existen justificadas razones por las cuales quienes trabajan en AI escojan lenguajes que reúnan ciertas características, pero la frase «lenguaje de AI» puede prestarse a confusión, por dos motivos. En primer lugar, sugiere que el LISP y el PROLOG pueden ser inadecuados para el procesamiento convencional de datos. En segundo lugar, y aún más importante, conduce a la suposición de que con el mero hecho de escribir un programa en LISP o en PROLOG uno ya se está introduciendo en el campo de la AI. Sugiere, asimismo, que otros lenguajes para procesamiento de símbolos, como pueden ser el POP-11 y el SNOBOL4, por nombrar sólo dos, son inadecuados para el trabajo serio en AI.

El LISP es un «veterano» entre los lenguajes de programación. Fue desarrollado inicialmente por John McCarthy en el MIT (Massachusetts Institute of Technology) a finales de los cincuenta y, en consecuencia, es tan antiguo como el COBOL. El lenguaje quedó estructurado en 1961, y desde entonces casi no ha experimentado modificaciones. Ya hemos ofrecido una breve serie de introducción al LISP, pero hay un concepto que no ha sido mencionado hasta ahora y que es de gran importancia para las aplicaciones de AI: la *lista de propiedades*.



Entorno de inteligencia artificial
La red INTERLISP-D de Rank Xerox conforma un entorno de programación completo para investigadores de AI, incluyendo gráficos interactivos, herramientas para depuración y grandes facilidades para almacenamiento en línea en forma de una unidad de disco de 29 megabytes. El sistema de programación del conocimiento LOOPS, utilizado en conjunción con INTERLISP-D, permite al programador de AI seleccionar una combinación de diferentes estilos de programación, tales como una codificación orientada hacia el objeto o una orientada hacia la regla, para construir sistemas de conocimiento de bajo costo

Cortesía de Artificial Intelligence Ltd.

Cada átomo posee su propia lista de propiedades, que se compone de pares de *valores de atributos*. La lista de propiedades es, en realidad, una descripción del átomo, y proporciona fácil acceso a una clase de estructura de base de datos. Las funciones GET y PUTPROP se utilizan para manipular listas de propiedades. Por ejemplo:

```
(PUTPROP 'AYER 0.1 'PRECIPITACIONES)
(PUTPROP 'MAÑANA 'NUBLADO
'PROBABILIDAD)
```

tiene el efecto de insertar 0.1 como el valor del atributo PRECIPITACIONES del átomo AYER, y de poner NUBLADO como el valor del atributo PROBABILIDAD para el átomo MAÑANA. Los atributos PRECIPITACIONES y PROBABILIDAD son como campos de un registro de base de datos. El valor de cualquier atributo dado se puede recuperar mediante el empleo de GET. Por ejemplo:

```
(SETQ LLUVIA (GET 'AYER 'PRECIPITACIONES)
```

obtendría el valor PRECIPITACIONES de la lista de propiedades de AYER y lo asignaría como nuevo valor para el átomo LLUVIA. (En realidad, el valor de un átomo no es más que un tipo especial de propiedad que está disponible sin necesidad de utilizar GET y PUTPROP.)

Las listas de propiedades se pueden emplear para construir estructuras de datos complejas y flexibles, y se han utilizado para modelar el funcionamiento de la memoria humana.

El PROLOG constituye la principal alternativa al LISP para la programación en AI. En Europa está alcanzando mayor difusión que el LISP, si bien éste aún prevalece en Estados Unidos. Debido a que el PROLOG es un lenguaje declarativo, permite que el programador especifique hechos y reglas acerca de objetos y relaciones. Estos hechos y reglas se pueden entonces utilizar para responder preguntas acerca de los objetos y las relaciones implicados.

En pocas palabras, el PROLOG responde preguntas mediante el uso de un método de provisión de teoremas basado en el *principio de resolución*, que niega la proposición a demostrar e intenta refutarla: una versión sofisticada de la técnica de la *reductio ad absurdum*. Pero en cuanto concierne al usuario, sólo está buscando en una base de datos.

El PROLOG, al igual que el LISP, posee facilidades para construir complejas estructuras de datos (incluyendo listas) y para entrada/salida, cálculos, ma-



nipulación de archivos, etc. En muchos sentidos no logra responder a los ideales de la programación lógica, pero aun así es potente y muy flexible.

No obstante, el LISP y el PROLOG no son las únicas herramientas de software adecuadas para la programación de AI. El POP-11 (especialmente en el entorno PROPLOG desarrollado en la británica Universidad de Sussex) también es un candidato para los trabajos de AI, y para programar AI es bastante posible utilizar lenguajes tradicionales tales como el C y el PASCAL (o incluso el BASIC). Ello, sin embargo, exige un mayor esfuerzo.

En razón de la complejidad de los problemas de AI, todo aquello que contribuya a facilitar la programación es siempre bien recibido. El tipo correcto de lenguaje es de gran ayuda, por supuesto, pero también cuentan otros muchos factores. Una tendencia moderna que se mantendrá sin ninguna duda es el uso de entornos de AI ejecutados en centros de trabajo de inteligencia artificial.

Ejemplos de tales entornos son el LOOPS y el POPLOG. LOOPS (LISP *Object-Oriented Programming System*: sistema de programación en LISP orientado hacia el objeto) se desarrolló en el Centro de Investigación Xerox de Palo Alto y se basa en un dialecto de este lenguaje denominado INTERLISP-D; pero es mucho más que un sistema de LISP. Proporciona múltiples herramientas de software, algunas convencionales (tales como administración de ventanas e iconos) y otras menos convencionales (como procedimientos de inferencia preempaquetados para uso en sistemas expertos). Todo el sistema opera en un centro de trabajo de AI específico que contiene un procesador optimizado para ejecutar LISP. El POPLOG se ejecuta en la gama de ordenadores VAX, de modo que no requiere hardware especializado; pero desde el punto de vista del software, se trata de una idea similar al LOOPS. Proporciona edición en pantalla y otros útiles paquetes estándares, así como la capacidad de llamar ya sea al LISP o al PROLOG desde POP-11 (su lenguaje principal), ofreciendo de este modo una positiva síntesis de ambos. Además, existe toda una biblioteca de emparejamiento de patrones y otros procedimientos elaborados específicamente para aplicaciones de AI.

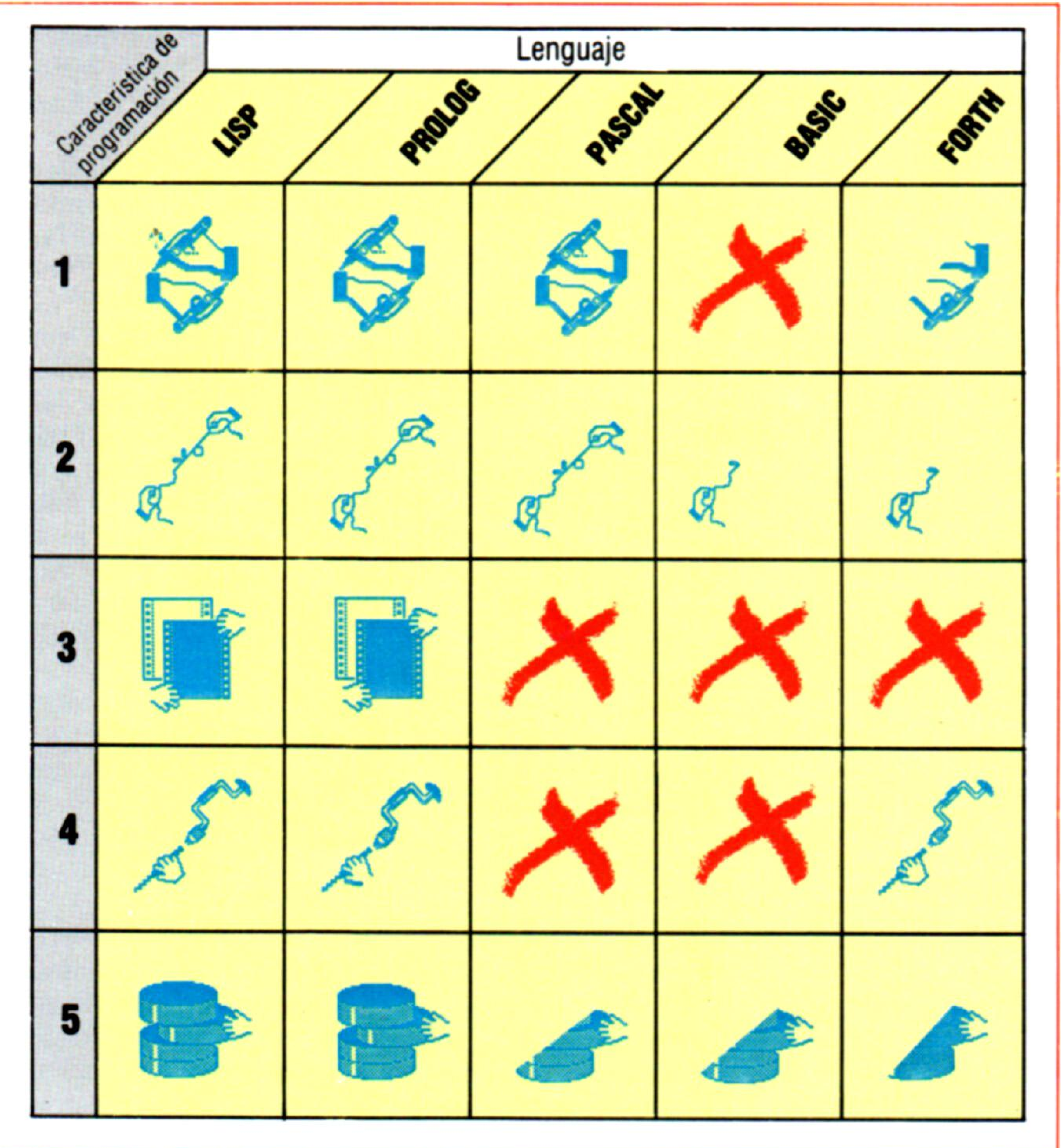
Por encima de todo, lo que ofrecen tales sistemas es un entorno de software muy favorable para la productividad del programador. Su principal desventaja es el aislamiento: se pueden desarrollar aplicaciones inteligentes con mucha rapidez, pero no se pueden ejecutar en el ordenador personal medio.

Al comenzar la programación, es muy raro que el problema de AI se comprenda en su totalidad. En realidad, los investigadores de AI con frecuencia escriben programas como un medio para obtener una visión más profunda de sus problemas. Por consiguiente, en la programación de AI es de capital importancia la *elaboración rápida de prototipos*, estilo de programación en el cual los sistemas evolucionan de forma gradual mediante muchas pequeñas adiciones y alteraciones. Las herramientas de software tradicionales y las metodologías de ingeniería de software no son adecuadas para satisfacer las exigencias de la programación de AI, la cual, como sabemos ahora, es una tarea difícil.

Aptas para el trabajo

Con el correr de los años se han desarrollado numerosas herramientas para facilitar la tarea del programador de AI; éstas poseen algunas importantes características en común.

1. El lenguaje debe soportar repetición.
2. El lenguaje debe disponer de buenas facilidades para manipulación de series y de símbolos, permitiendo la construcción de estructuras de datos flexibles de complejidad arbitraria.
3. Es importante que exista uniformidad de programa y datos: tanto el LISP como el PROLOG representan programas y datos con el mismo formato.
4. La sintaxis ha de ser ampliable: como en el LISP y en el PROLOG, debe ser posible la construcción de un nuevo lenguaje además del original.
5. El acceso a una base de datos incorporada de alguna clase es crucial: el PROLOG posee en su núcleo lo que representa una base de datos relacional, pero el LISP (con listas de propiedades) también ofrece tal facilidad. Además, en los trabajos de AI se necesitan todas las otras herramientas de trabajo que contribuyen a la simplificación de la programación (editores, rutinas para gráficos, verificadores de sintaxis, depuradores, compiladores, productores de documentación, etc.), al menos en la misma medida que en otras áreas de la informática. El gráfico refleja la distribución de estas características entre cuatro diferentes lenguajes de alto nivel



Ian McKinnell

Mundo de palabras

En FORTH, los programas se construyen a partir de palabras que puede definir el mismo programador

Es agradable que todas las instrucciones se utilicen de la misma manera, tanto si están incorporadas en el lenguaje como si se definen luego como parte de un programa. De modo que el FORTH tiene todas sus instrucciones incluidas en un diccionario. Al igual que un diccionario normal, éste consiste en una lista de palabras y sus definiciones, si bien no se almacenan por orden alfabético, sino por el orden según el cual se van definiendo. Cada definición le dice al ordenador exactamente qué hacer cuando se usa esa palabra, ya sea porque se ha digitado directamente (de forma interactiva) o bien porque forma parte de la definición de otro vocablo que se esté utilizando.

En FORTH, todo lo que el sistema reconozca es una palabra, con una definición en el diccionario. Las únicas excepciones a esta norma son los números, que se reconocen como tales de la forma habitual. Algunas palabras se definen como rutinas (con dos puntos o un punto y coma como parámetros), mientras que otras se definen como variables o constantes; pero todas se almacenan en el mismo diccionario. Una palabra puede, en consecuencia, desempeñar varios papeles diferentes, tal como veremos a continuación.

Las instrucciones incorporadas en el sistema son palabras. Por ejemplo, la instrucción ORDS (muchos sistemas utilizan, en cambio, el VLIST, abreviación de *Vocabulary* LIST, lista de vocabulario) visualiza una lista de palabras del diccionario. En algún lugar de la lista puede verse el vocablo WORDS (palabras). Las instrucciones que se le proporcionen en un FORTH ampliado también son palabras, como GRADOS y ELEVACION en el FORTH para telescopios que ya hemos analizado, o como FORWARD y RIGHT en el FORTH para tortugas.

Las instrucciones nuevas que añada el usuario son palabras, como ESTACIONAR y CUADRADO, y se definen entre un signo de dos puntos y un punto y coma, como en:

```
:ESTACIONAR 0 GRADOS ACIMUT 90 GRADOS
ELEVACION;
```

Como puede observar, primero va el signo :, luego la palabra que se esté definiendo, luego la definición y por último el ;. Cuando todo esté incluido en una misma línea, es muy importante asegurarse de que todos los componentes estén separados por espacios. Por otra parte, pueden dividirse en varias líneas, lo que suele resultar más legible:

```
:ESTACIONAR
  0 GRADOS ACIMUT
  90 GRADOS ELEVACION
;
```

Todo el programa es una palabra. De hecho, el FORTH realmente no piensa que nada sea *el* programa. Si las definiciones componen una única instrucción grande (que podría llamarse RUN si así se deseara), entonces podría pensar en ella como si fuera el programa, utilizando las otras palabras como subrutinas; pero no existe razón alguna por la cual no puedan tenerse en el diccionario otras instrucciones/programas independientes al mismo tiempo, con nombres diferentes. Las subrutinas también son palabras. No son otra cosa que instrucciones que se emplean en otra definición.

El FORTH no hace distinción entre instrucciones, programas y subrutinas. Todas ellas se definen utilizando : y ;, y todas se pueden emplear ya sea directamente desde el teclado o bien indirectamente desde otra definición. Debido a que estas definiciones utilizan un signo de dos puntos, se las llama *definiciones de dos puntos*.

Las variables también son palabras. Usted declara una variable (LONGITUD, p. ej.) digitando:

```
VARIABLE LONGITUD
```

La palabra LONGITUD posee, entonces, una definición en el diccionario que incluye algo de espacio de memoria para su valor. Usted debe declararla de este modo antes de poder utilizarla. Si no lo hace, simplemente no habría ninguna definición de ella en el diccionario y no se la reconocería como variable. Puede utilizar variables con los caracteres @ (que significa *traer*) y ! (*almacenar*):

```
LONGITUD @
```

le da el valor de LONGITUD, y:

```
26 LONGITUD !
```

establece su valor en 26 (como en LET LONGITUD = 26 en BASIC).

Es fácil olvidarse del @, pero usted debe utilizarlo para obtener el valor de la variable. Sin él obtendrá la *dirección* de la variable en lugar del valor que podría tener asignado.

Las constantes también pueden ser palabras,

Números como palabras

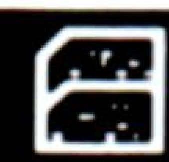
En la definición:

```
0 CONSTANT 0
```

0 previamente era un número que se podía reconocer en función de la regla 2. A partir de ahora tendrá una definición en el diccionario y, por tanto, será reconocido en función de la regla 1 antes de ser examinado por la regla 2. Al estar definido como el número 0, no hay ninguna diferencia obvia, pero es posible que en el FORTH suyo sea más eficaz de esta forma; de hecho, el figFORTH define al 0, 1 y 2 en el diccionario por la enorme frecuencia con la que se utilizan. Una definición más equívoca sería:

```
12 CONSTANT 13
```

de modo que cuando usted digite 13, en realidad obtendrá 12. (Sin embargo, podría seguir obteniendo el auténtico 13 digitando 013.)



como parece natural. Si emplea repetidamente un número con un cierto significado, quizá prefiera otorgarle un nombre significativo definiendo una palabra como constante. Por ejemplo:

```
66 CONSTANT CLIQUITICLIC
```

A partir de entonces la palabra CLIQUITICLIC significa simplemente 66. Por supuesto, podría haber utilizado en cambio una variable, pero la ventaja de una constante es que no se necesita el @. Evidentemente, con ella usted tampoco puede emplear la instrucción de almacenar (!).

Observe que no puede utilizar declaraciones VARIABLE y CONSTANT dentro de una definición de dos puntos; más adelante verá el motivo cuando analicemos cómo se almacenan las definiciones en el diccionario. Sin embargo, resulta tentador intentar definir palabras como:

```
:INICIALIZAR
   VARIABLE MARCADOR   0 MARCADOR!
   VARIABLE METAS   0 METAS!
;
```

pero no funcionará. Otra consecuencia de esto es que todas las variables son *globales*, como en BASIC. Las variables locales, como las que encontrará en PASCAL, no existen en FORTH.

El FORTH incluye +, −, *, / y muchos otros operadores aritméticos. Como todas las palabras, deben ir separados de otras palabras mediante espacios, y esperan hallar sus argumentos ya allí. De modo que se escribe:

```
2 3 +
```

en lugar de 2 + 3. Este punto lo explicaremos en mayor profundidad en el próximo capítulo, con el funcionamiento de la «pila».

Los dos puntos, VARIABLE y CONSTANT son inusuales porque introducen nuevas definiciones en el diccionario; por este motivo se las denomina *palabras definitorias*. Pero, exceptuando esto, continúan siendo palabras del diccionario, iguales a cualquier otra. En FORTH realmente es posible que uno mismo defina nuevas palabras definitorias.

Los símbolos de estructura del programa incluyen DO y LOOP, que ya hemos visto anteriormente (similar al FOR...NEXT del BASIC), ; (el final de una definición de dos puntos) y otras estructuras de programa como IF...THEN y BEGIN...UNTIL, que se utilizan para bifurcaciones y bucles. Nuevamente tiene usted la posibilidad, aunque no siempre es fácil, de definir sus propias palabras nuevas de estructura de programa.

Dado que ya sabemos lo que ocurre en el diccionario, podemos ver cómo trata el FORTH nuestra entrada. Trata a cada grupo de símbolos sin espacios como una palabra potencial y luego procede de acuerdo con tres reglas principales.

1. Si encuentra la palabra en el diccionario, lleva a cabo lo que indica la definición. Si estuviera definida más de una vez, se utilizaría la definición más nueva.
2. Si la palabra no está en el diccionario, el FORTH comprueba si el símbolo (o grupo de símbolos) es un número y, si así es, lo recuerda temporalmente. (Lo coloca en un trozo de memoria llamado *pila*).
3. Si se encuentra con que el grupo de símbolos no es un número ni está en el diccionario, o sea irreconocible, el FORTH así se lo hace saber.

Palabras y espacios

Una palabra es, simplemente, cualquier secuencia de caracteres. En consecuencia, en FORTH las siguientes son todas palabras (si bien no necesariamente definidas en el diccionario):

```
SALCHICHA
Tubo de gomadesemulsionada
239
pH
25p
+
«merguez»
,!XXX)luego-allíQWERTY
```

El FORTH no realiza el más mínimo intento por otorgarle un significado especial a los caracteres especiales, exceptuando el espacio. Probablemente usted elija palabras como LONGITUD y MARCADOR para sus variables, pero el FORTH no se preocupará en absoluto si usted decide llamarlas 23PI/2 o FORI=1TO10. Quizá se pregunte por qué otros lenguajes son mucho más restrictivos. En muchos, el nombre de una variable, por ejemplo, debe ser una letra seguida de otras letras cualesquiera o dígitos. La respuesta en realidad reside en la actitud de ellos hacia los espacios. La mayoría de los lenguajes tratan los espacios como una especie de elemento decorativo. Pero en esos lenguajes puede omitirlos si así lo desea, y probablemente así lo hará al escribir expresiones: en BASIC normalmente no introducirá espacios en la expresión 2 + 3. Los lenguajes, por tanto, deben ser capaces de distinguir entre los diversos componentes del programa. Si usted pudiera denominar 2 + 3 a una variable en BASIC, el intérprete de BASIC no sabría si eso significaría el nombre de la variable o una expresión; de modo que debe restringir los nombres posibles de las variables para que no se produzcan tales ambigüedades. El FORTH asume un enfoque diferente. Utiliza los espacios de forma inflexible, como separadores entre palabras, y ésa es la única regla que necesita. (Éste es un punto crítico del FORTH y, por consiguiente, al digitar listados se debe tener especial cuidado con los espacios.)

Complementos al FORTH

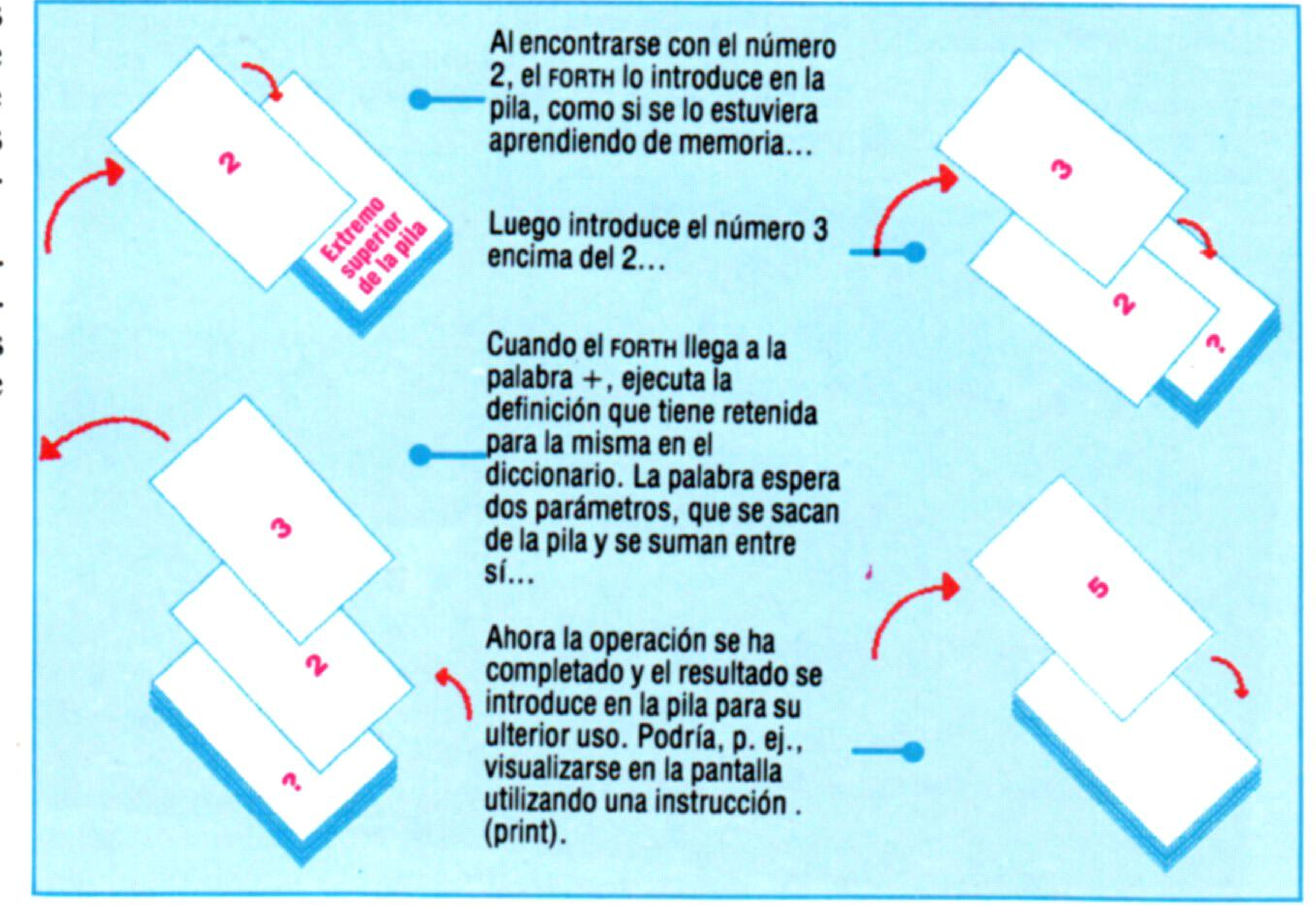
En figFORTH, VARIABLE debe ir precedida por un número:

```
10 VARIABLE
   BOTELLASVERDES
```

es el valor con el cual ha de comenzar la variable

Todo tiene sentido
Cuando el FORTH encuentra una palabra, procede de acuerdo con la definición que tiene almacenada en su diccionario para dicha palabra. Sin embargo, cuando halla un número, lo coloca en una zona de la memoria conocida como *pila*. El número permanece allí hasta que posteriormente el FORTH necesite «recordarlo» (en el curso de la ejecución de la definición de una palabra, p. ej.). Aquí vemos lo que sucede cuando el FORTH interpreta la entrada:

2 3 +



Estas reglas han de modificarse ligeramente en cuanto a las definiciones de dos puntos, porque las acciones apropiadas no se llevan a cabo inmediatamente, sino que se recuerdan en la definición.

Las excepciones obvias las constituyen las palabras (como :, VARIABLE y CONSTANT) y éstas provocan cierta dificultad. Cuando usted escribe:

```
VARIABLE MARCADOR
```

parecería generar un error de la regla 3. Pero en realidad VARIABLE se ocupa de MARCADOR y las tres reglas no la llegan a ver nunca. De lo contrario, nunca podríamos entrar palabras en el diccionario.

Se puede apreciar cómo el diccionario confiere al FORTH su capacidad de interacción y su ampliabilidad. La interactividad está asegurada en función de las tres reglas, y la ampliabilidad se debe al hecho de que todas las definiciones nuevas se incluyen en el diccionario en términos de igualdad respecto a las antiguas.

De vuelta al cuadrado uno

La experiencia en el campo de la programación en código máquina puede facilitar la comprensión del FORTH, pero para quienes carezcan de tal experiencia presentamos el siguiente ejemplo comentado. Para definir una palabra CUADRADO como una subrutina que dibuje un cuadrado en la pantalla con lados de longitud LADO, procederíamos de la siguiente manera. En primer lugar, hemos de declarar la variable LADO:

```
VARIABLE LADO
```

Ésta le indica al FORTH que reserve un espacio en la memoria donde se pueda almacenar un valor numérico. Podemos desplazar valores desde y hacia esta dirección mediante el empleo de @ (traer) y ! (almacenar). De modo que p. ej.:

```
20 LADO !
```

le asignaría a la variable LADO el valor 20.

Ahora podemos definir nuestra nueva palabra CUADRADO utilizando las «palabras»: y; (tal como emplearíamos TO y END en LOGO):

```
:CUADRADO
  LADO !
  4 0 DO
    LADO @ ADELANTE
    90 DERECHA
  LOOP
;
```

Este procedimiento espera dos cosas. Primero, que ya se hayan definido las palabras DERECHA y ADELANTE y que ofrezcan facilidades de tortuga como en LOGO, de modo, que, por ejemplo:

```
50 ADELANTE
```

haría que la tortuga avanzara 50 unidades de pantalla hacia adelante. En segundo lugar, se espera que el usuario entre un valor para LADO, de modo que entrando:

```
50 CUADRADO
```

se dibujaría un cuadrado con lados de 50 unidades.

Trabajemos ahora con la subrutina y veamos exactamente cómo funcionaría si en realidad hubiéramos entrado la instrucción 50 CUADRADO. El FORTH examina su entrada y encuentra que el primer grupo de símbolos está separado por espacios, en este caso 50. El FORTH comprueba entonces si este grupo ya se ha entrado en su diccionario. Suponiendo que usted no haya definido con anterioridad el grupo de símbolos de caracteres 50 como una palabra, el FORTH no logrará hallarlo en el diccionario y comprobará si se trata de un número, como por supuesto sucede en este caso. Por consiguiente, el FORTH toma el número, lo coloca en un almacén de la memoria y comprueba el siguiente grupo de símbolos, que es CUADRADO. Evidentemente, CUADRADO sí está en el diccionario (acabamos de definirlo), de modo que el FORTH recupera su definición y procede a evaluarla. Los primeros términos con los que se tropieza son LADO! En este punto, usted bien podría pensar que algo funciona mal, puesto que LADO! no parecería tener ningún valor que asignarle a LADO; pero el FORTH comprueba su almacén de memoria y encuentra allí el valor que entramos (50) y lo utiliza, asignándole, por tanto, 50 a la variable LADO. Observe que el FORTH, a diferencia del LOGO, no comprueba esto: se limita a dar por sentado que usted ha entrado un número en el lugar apropiado, de modo que si usted simplemente llamara a la subrutina CUADRADO con CUADRADO en lugar de 50 CUADRADO, obtendría algunos resultados impredecibles y no un mensaje de error. 4 0 DO...LOOP se explica bastante por sí solo, pero es interesante observar cómo dentro de ese bucle se pasa el valor de la variable LADO a la subrutina ADELANTE mediante las palabras LADO @ ADELANTE. Éstas primero llaman a la dirección de LADO y luego (utilizando @) traen un valor desde esa dirección, listo para que ADELANTE lo utilice. (Compare esto con el método empleado por la rutina CUADRADO, que utilizaba DUPLICAR para presentar un valor para ADELANTE.) Los términos 90 DERECHA, por supuesto, son necesarios para hacer que la tortuga gire 90 grados tras dibujar cada lado, completando de ese modo el cuadrado. Puede ser que a primera vista el FORTH parezca algo confuso, en especial si usted no es todavía un programador experimentado. No obstante, en su estructura y comportamiento es muchísimo más directo de lo que sugiere en apariencia



El poder de la palabra
Definir una palabra CUADRADO ilustra únicamente un aspecto del potencial de ampliación que posee el FORTH. No sólo es posible construir «diccionarios» de palabras para controlar dispositivos mecánicos, como el telescopio que vemos en la fotografía, sino también idear sistemas de control para aplicaciones más abstractas. Por ejemplo, una casa productora de software británica (Mastertronics) utiliza el FORTH, en la creación de juegos de aventuras, para definir palabras que manipulen personajes, objetos y otras configuraciones propias de los entornos de fantasía



Plus ça change

El BBC+ obvia los principales puntos débiles de su antecesor, pero su precio le impide acceder a un mercado definido

Cuando se le concedió a Acorn el contrato del BBC Microcomputer, en 1981, la máquina era considerada por la mayoría de la industria como un inmenso adelanto en materia de ordenadores personales. Su velocidad y su versatilidad hacían que superara en rendimiento a todos los otros micros personales existentes en el mercado. Sin embargo, desde entonces, el mercado ha cambiado sustancialmente. Aunque la gama de interfaces y periféricos para el BBC Micro sigue siendo incomparable, con el correr del tiempo el ordenador ha dejado traslucir algunas debilidades.

El principal punto débil, que ha marcado al ordenador desde sus comienzos, ha sido la relativa escasez de memoria disponible. Algunas de las modalidades para gráficos de mayor resolución dejan muy poca RAM al usuario para el desarrollo de programas. Ello no se debe a que el BBC Micro carezca de RAM para el usuario (posee los mismos 64 k de otros muchos micros personales), sino a que las avanzadas configuraciones para gráficos del ordenador no caben con comodidad en los confines de las capacidades para direccionamiento de memoria de un procesador de ocho bits.

Hacia 1984 este problema representó una amenaza aún mayor para el futuro a largo plazo de la máquina. Mientras otros fabricantes sacaban partido de la caída del costo de los microprocesadores en el mercado internacional (y en particular de los chips de memoria) para recortar sus precios, Acorn mantuvo inalterado el precio del BBC Modelo B. Acorn pudo hacerlo en parte gracias al valor del prestigio del contrato con BBC, y también en parte al subsidio que concedió el gobierno británico a los establecimientos educativos que adquirieran el ordenador a través del Microcomputer Educational Programme (MEP). Además, mientras los otros fabricantes, como Sinclair, estaban rediseñando sus placas de circuito impreso para sacar ventaja de los chips que estaban apareciendo, de mayor capacidad y menor precio, Acorn no consiguió hacerlo. Ello determinó que el costo de producción del BBC Micro alcanzase un nivel considerablemente superior al de todos sus competidores.

Por último, la tecnología de ocho bits del BBC Micro corría el peligro de quedar anticuada. Sinclair Research, uno de los mayores rivales de Acorn para el contrato con BBC, no tuvo reparos en dar publicidad al hecho de que su ordenador de 16 bits, el QL, estaba diseñado para competir exactamente al mismo precio que el BBC Micro.

Desde entonces, el peligro de que los procesadores de 16 bits conviertan en obsoletos a sus equivalentes de ocho bits ha disminuido, aunque de forma temporal. El repentino descenso en las ventas de ordenadores y la crisis financiera tanto de Acorn como de Sinclair han hecho que la industria informática actúe con mayor cautela. El resultado ha sido que en lugar de introducir máquinas nuevas con todo el margen de riesgo que ello implica, casi todos los principales fabricantes de micros personales hayan optado simplemente por perfeccionar sus productos existentes. Para ser justos, esto responde a los deseos del público. El mercado se está volviendo mucho más sofisticado y el cliente no parece mostrarse interesado por las últimas innovaciones en materia de hardware y sí, en cambio, por la calidad y cantidad de la base de software. Y de allí el desarrollo de ordenadores más potentes compatibles con las bases de software desarrolladas para máquinas anteriores.

La base de software

El BBC + es una respuesta de Acorn a la demanda de los clientes que requieren grandes bases de software ya existentes, pero también mayor memoria para ejecutar programas más largos y sofisticados. El ordenador está equipado con 32 k más de RAM que su predecesor. Al estar basado en un procesador de ocho bits, la cantidad máxima que se puede direccionar directamente se limita, como es obvio, a 64 K. En consecuencia, al igual que Atari y Commodore, Acorn ha utilizado la técnica de *conmutación de bancos*, que permite acomodar las zonas de memoria adicionales.

La técnica de conmutación de bancos permite que el procesador «mire» una de dos zonas de memoria. Se acomodan dos bancos de memoria de modo que ocupen las mismas direcciones de memoria. Por cuanto concierne al procesador, sólo hay una única posición de memoria; pero, según en qué zona de memoria está «depositada», la dirección puede contener varios contenidos diferentes.

Los 32 K de RAM adicionales se hallan en las direcciones 12288 (hexadecimal 3000) y 45055 (hexadecimal AFFF). Ocupa la mayor parte de la zona para programas en BASIC y la zona de memoria ocupada por las ROM paginadas (las ROM paginadas son las ROM de BASIC y aquellas que puedan haber instalado los propios usuarios, como View y LOGO).

La RAM en sombra propiamente dicha está dividida entre estas dos secciones, que vamos a examinar por separado. Como ya hemos visto, el mayor problema del espacio de memoria en el BBC Micro es la gran cantidad de RAM que se requiere para soportar una pantalla de gráficos en alta resolución. Por tanto, la enorme masa de RAM adicional se ha cedido completamente para usarla como RAM de video. De hecho, los 20 K completos que «ensombrecen» la zona para programas en BASIC queda reservada para esta finalidad, lo que representa memoria suficiente para soportar la pantalla más detallada. Al rediseñar la placa de circuito impreso con el objeto de dar cabida a los chips de RAM adicio-

BBC+

MEMORIA

76 Kbytes de RAM, de los cuales hay 64 Kbytes disponibles para programas en BASIC, 32 Kbytes de ROM, ampliables a 192 Kbytes

CPU

Procesador 6512 trabajando a 2 MHz

DISCOS

El BBC+ viene equipado con un sistema DFS Acorn como estándar, si bien las unidades de disco son adicionales

DOCUMENTACION

Se ha actualizado la guía para el usuario del BBC Micro, de por sí ya muy completa, para incluir información adicional sobre la RAM en sombra y otros puntos útiles

VENTAJAS

La provisión de hardware extra del BBC+ es una respuesta a muchas de las críticas que suscribió el BBC Micro original. Ahora se dispone de suficiente memoria como para permitir que los programadores soporten una pantalla de alta resolución a la vez que escriban sofisticados programas en BASIC

DESVENTAJAS

A pesar de ser una máquina mejorada, el BBC+ sigue siendo caro para la mayoría de los usuarios de micros personales. En la actualidad el mercado educativo parece estar optando por máquinas MS-DOS, lo que restringe aún más los posibles mercados para el BBC+

nales, Acorn ha tenido oportunidad de introducir algunos otros cambios. Uno de los más notables es que la nueva máquina opera con un procesador 6512 en vez de con un 6502. Con ello se estandariza parte del intercambio de información entre la CPU y el chip para periféricos 6522 con el que Acorn tuvo algunos problemas en el pasado. La dirección de conmutación de bancos propiamente dicha (hexadecimal FE34) está contenida en una de las ROM (IC36) del sistema operativo que se han vuelto a diseñar. Puesto que el BBC Micro opera en gran medida en base a interrupciones, la adición de esta dirección no ha requerido ningún cambio sustancial en el sistema.

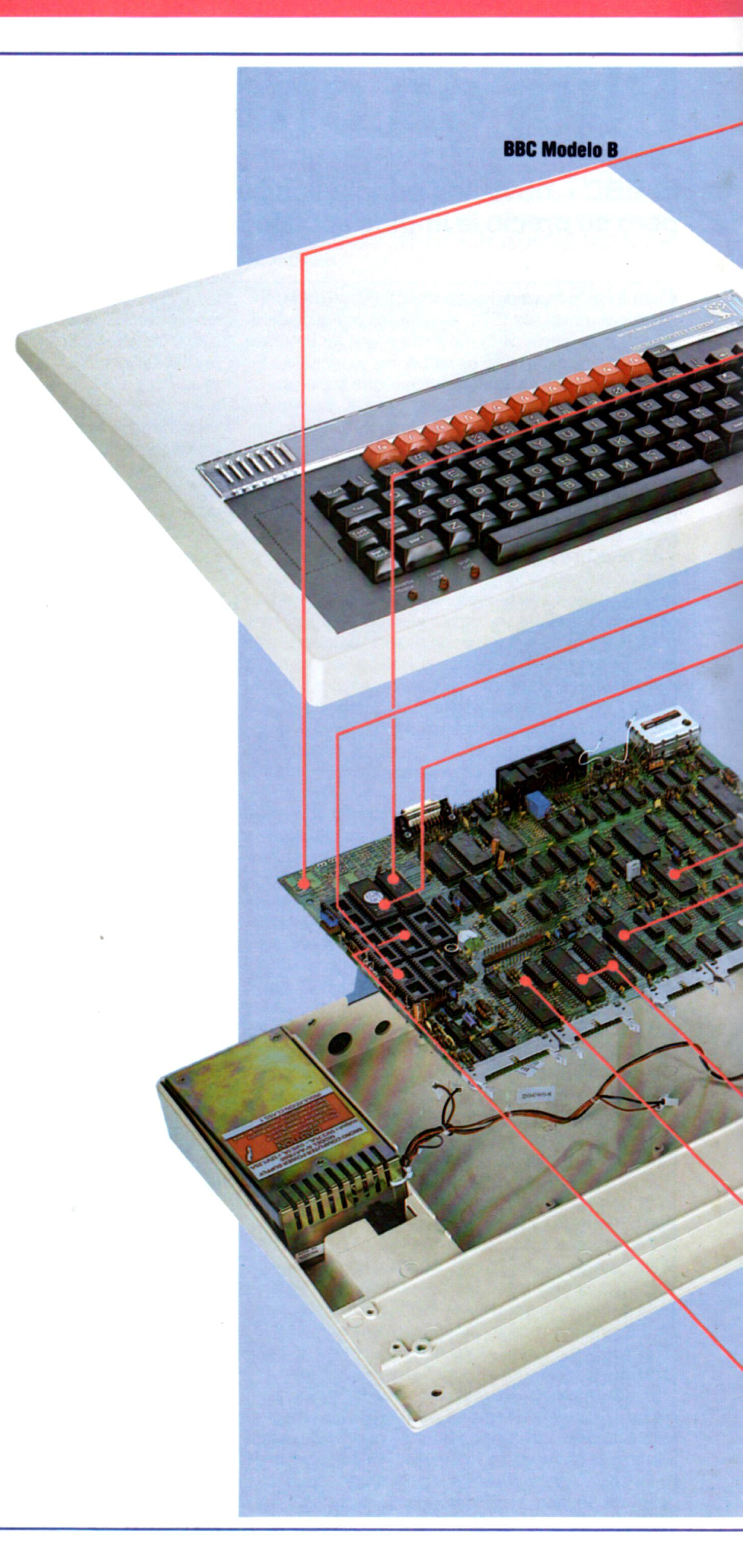
Los 12 K restantes de memoria que se han proporcionado al BBC + se hallan en la zona de direcciones de memoria entre 32768 (hexadecimal 8000) y 45055 (hexadecimal AFFF). Ésta es la zona reservada para uso de las ROM paginadas. Nuevamente la conmutación de bancos permite que las ROM y la RAM en sombra de abajo compartan el mismo tiempo.

BASIC y OS rediseñados

Los usuarios del BBC + tienen una ventaja sobre sus compañeros de Atari y Commodore, quienes deben utilizar instrucciones POKE para poder acceder a los bancos de memoria adicionales, porque Acorn también ha vuelto a diseñar su BASIC y sus ROM MOS (machine operating system: sistema operativo de la máquina), para proporcionar apoyo de software para las facilidades de conmutación entre bancos.

A la memoria de pantalla en sombra se puede acceder de varias formas. Utilizando la instrucción MODE, seguida de un número entre 128 y 135, se producirá una pantalla en sombra por defecto correspondiente a las ocho modalidades de pantalla entre 0 y 7. Sin embargo, las subsiguientes instrucciones MODE que no estén comprendidas entre 128 y 135, darán por defecto la zona de RAM normal, lo que obviamente constituye un error fatal. La instrucción *SHADOW fijará permanentemente la pantalla en la RAM en sombra, impidiendo que el usuario escriba sobre cualquier programa que hubiera en BASIC. Para salir de esta situación, se ha de ejecutar la instrucción *SHADOW 1. Naturalmente, como todas las instrucciones * BBC, están las correspondientes instrucciones *FX; en este caso, *FX114 y *FX114,1, respectivamente. De modo similar, las instrucciones MODE se pueden suplantar por VDU 22, (128 + n), donde n representa la modalidad requerida.

Al perfeccionar el BBC Micro, Acorn ha optado asimismo por incorporar en el equipo estándar un sistema de archivo en disco (DFS). Éste es útil para quienes deseen adquirir un ordenador y pretendan utilizar desde el principio unidades de disco; la alternativa consiste en instalar un DFS por un recargo adicional. Pero para la mayoría de los usuarios, que sólo desean utilizar con la máquina una unidad de cassette, esta provisión adicional se limita a incrementar el precio de partida del BBC +. Esto parece lamentable, dado que el costo extra tampoco hará nada por silenciar las críticas de quienes sostienen que la gama de micros BBC tiene fijado un precio excesivo.





BBC+

Placa de circuito impreso
Aunque del mismo tamaño, la placa de circuito impreso del BBC+ está empaquetada con menor densidad que la de su antecesor, reflejando la mayor capacidad de empaquetamiento de los chips que se fabrican desde 1981

ROM del sistema operativo
Al igual que la ROM de BASIC, se ha modificado para manipular la memoria adicional que se ofrece con el BBC+

Conectores para ROM de sintetizador de voz

BASIC ROM
Si bien el BASIC que se utiliza en ambas máquinas es compatible, el BBC + dispone de instrucciones extras para la RAM «en sombra» adicional

Controlador de video

Unidad central de proceso
El BBC Modelo B emplea como CPU un procesador 6502 estándar. Sin embargo, Acorn ha mejorado el BBC+ con el chip 6512, que posee mejores facilidades para comunicaciones

Chips RAM
A pesar de que el BBC+ tiene instalados 32 Kbytes de RAM adicionales, en realidad necesita menos chips para obtener la memoria extra

Chips PIA 6522

Archivo en disco
En el BBC Micro original había que instalar una gran cantidad de chips (que vemos aquí en forma de conectores vacíos) antes de poder utilizar una unidad de disco. Esto se ha reducido a apenas cinco chips

Ranura para ROM adicional
El BBC+ posee una ranura para ROM extra, que permite la instalación de otra aplicación basada en ROM

Primeras palabras

Iniciamos una nueva serie, en la que esbozaremos los requisitos para construir un tester digital sencillo. En esta introducción consideraremos algunos de los problemas que implica la conversión de analógico a digital y desarrollaremos una estrategia para obtener el diseño adecuado

Por lo general los ordenadores operan de forma totalmente digital. Incluso las interfaces «humanas» de la entrada por teclado y la salidad de video se basan exclusivamente en sistemas de circuitos digitales. La utilidad de los ordenadores se podría ampliar considerablemente si pudieran comunicarse con el «mundo exterior». Pero éste tiende a ser un mundo analógico de voltajes, temperaturas, alturas, pesos, etc., infinitamente variables. En consecuencia, cuando los ordenadores necesitan recoger datos o controlar dispositivos del mundo real, se hace necesaria la conversión entre un entorno analógico y uno digital.

Un ordenador unido a un dispositivo sensible a la temperatura, por ejemplo, habrá de trabajar con números binarios que correspondan a las temperaturas medidas con el fin de decidir lo que debe hacer. Ello implicaría lo que se conoce como un convertidor de analógico a digital, o A/D. Más adelante veremos algunas de las técnicas que se emplean para convertir mediciones analógicas (continuamente variables) en representaciones digitales (binarias).

El contrario de esta situación se produce cuando se utiliza un dispositivo digital, como puede ser un ordenador, para alterar el valor de algo del mundo real. A modo de ejemplo, consideremos un ordenador que se ha programado para producir música. El ordenador crea un valor digital binario, pero éste se ha de convertir en una frecuencia acústica. En este tipo de situaciones se requiere la conversión de señales digitales a señales analógicas.



El mercado del tester

Comparemos las características que ofrecen tres testers digitales típicos:



Lascar LMM 100 Bench Meter

Visualización:	LCD de 3½ dígitos
Escala:	25 escalas
Precisión:	−0,1% (voltios CC)
Potencia:	Pila
Otras características:	Indicaciones de pausa, polaridad, pila y superación de escala

Soar ME-531 Autoranging Hand-held Meter

Visualización:	LCD de 3½ dígitos
Escala:	Escala automática
Precisión:	0,8% (voltios CC)
Potencia:	Pila
Otras características:	Polaridad, indicación de sobrecarga, comprobación de diodos

Maplin Precision Gold M-5010 Basic Hand-held Meter

Visualización:	LCD de 3½ dígitos
Escala:	29 escalas
Precisión:	0,25% (voltios CC)
Potencia:	Pila
Otras características:	Indicación audible de continuidad, indicación de superación de escala y polaridad, comprobación de diodos

Marcus Wilson-Smith

Medidores digitales

Un multímetro o tester es un instrumento para medir la conductividad (medida en ohmios), la corriente (medida en amperios) y la tensión (medida en voltios) de un sistema eléctrico. En esta serie construiremos un tester digital, que efectuará estas mediciones con un elevado nivel de precisión y visualizará los resultados digitalmente mediante una visualización LCD o LED. El desarrollo de tal dispositivo sería una ardua empresa si no fuera por el advenimiento de los circuitos integrados a gran escala especializados, que combinan muchas etapas analógicas y digitales en un único chip. Al desarrollar nuestro DVM (*digital voltmeter*: voltímetro digital) deseamos diseñar un circuito que pueda trabajar a la vez como tester digital independiente, y también como interface directamente con un ordenador, para que éste pueda leer y procesar la tensión y otras magnitudes.

Un voltímetro digital utiliza una aguja que se desplaza por encima de una escala para proporcionar una lectura directa en voltios, amperios u ohmios. Combinaciones de resistencias en derivación y en serie les permiten a tales testers dar lecturas en términos de tensión, intensidad o resistencia, pero fundamentalmente todos trabajan de la misma manera. En definitiva, una corriente fluye a través de una bobina suspendida en el interior de un campo magnético, y la intensidad de la corriente determina hasta dónde girará la bobina contra la fuerza de un muelle. Una aguja conectada a la bobina muestra hasta dónde ha girado la misma, y la escala se calibra en términos de las unidades que se estén midiendo.

Cuando se ha de medir y visualizar digitalmente una tensión desconocida, los problemas son mucho mayores. Antes de pasar a ver cómo se realiza esto, es interesante considerar el problema inverso, el de convertir valores digitales en valores analógicos.

La conversión de digital a analógico (D/A) es una operación bastante directa. Supongamos que usted desea convertir una palabra de ocho bits (una unidad compuesta por ocho dígitos binarios) en una tensión analógica entre 0 V y 1 V. Una palabra de ocho vits puede representar cualquier valor entre 0 y 255 (binarios 00000000 y 11111111). Esa escala de un voltio se puede resolver, por lo tanto, en pasos de voltaje de 256 puntos, o 0,0039 voltios.

Los valores binarios se pueden convertir en valores analógicos utilizando las señales binarias para establecer corrientes cada vez más intensas. Los diagramas muestran tanto el tipo de convertidor D/A más simple, como el tipo más común, el R2R. En ambos casos, los interruptores mecánicos representan interruptores electrónicos activados mediante señales binarias, significando «cerrado» el 1 binario y «abierto» el 0 binario. Cada dígito binario positivo activa un interruptor electrónico que simplemente permite que fluya más corriente.

La conversión de señales analógicas (p. ej., tensiones) en sus equivalentes digitales es bastante más difícil, y para ello hay muchas técnicas disponi-



bles. Al igual que en la conversión D/A, el número de bits utilizado determina la resolución de la medición; no tiene nada que ver con la gama de tensiones que puede medir. Para una mayor resolución serán necesarios más bits, pero para la finalidad de nuestro DVM, ocho bits serán más que adecuados.

Nuestro diseño tendrá una gama básica de 0 V a 2 V, por lo que con los ocho bits de resolución, los intervalos medibles serán de 0,0078 voltios. Los márgenes de medida de 0 V a 20 V y de 0 V a 200 V se conseguirán utilizando un sencillo circuito divisor de potencial constituido por resistencias.

Circuitos de conversión

La conversión de analógico a digital es bastante más complicada y cara que la conversión D/A, y en parte depende de cuán rápidamente deba realizarse la conversión y de cuántos bits de resolución se necesiten. Si hay que convertir señales de audio de márgenes de frecuencia de hasta 15 kHz en señales analógicas, por lo menos será necesario convertir 30 000 muestras de la señal cada segundo. Si hay que digitalizar señales de vídeo, la velocidad de muestreo deberá ser mucho más elevada.

Afortunadamente, un DVM sólo tiene que medir tensiones continuas (CC) o alternas (CA) de baja frecuencia, por lo que bastará una baja velocidad de conversión; sin embargo, para una precisión razonable, serán precisos ocho bits de resolución.

Sencillo convertidor D/A
Un convertidor de digital a analógico de resistencia ponderada utiliza resistencias en potencias crecientes de dos. En un circuito verdadero, los interruptores serían tipos analógicos transistorizados activados por una señal «cero» o «uno». Cuanto más resistencias haya en el circuito, más corriente circulará por el punto sumador del amplificador operacional, que convierte la corriente en una tensión de salida proporcional

Convertidor de tipo R2R
Los convertidores D/A suelen emplear el circuito de resistencias denominado R2R. La corriente se divide por dos cuando pasa por el atenuador escalonado, y el valor establecido por los interruptores se vuelve a relacionar como potencias de dos

Sencillo convertidor A/D
La señal del reloj hace que el contador de cuatro bits se incremente a partir de cero, y el convertidor D/A convierte su salida binaria en una tensión de salida equivalente que se aplica a un comparador. Cuando la tensión de salida se corresponde con el voltaje de entrada que se está midiendo, el comparador produce una salida que detiene el contador. Entonces, el valor de ésta se puede comprobar, calculando el valor de la tensión equivalente

Kevin Jones

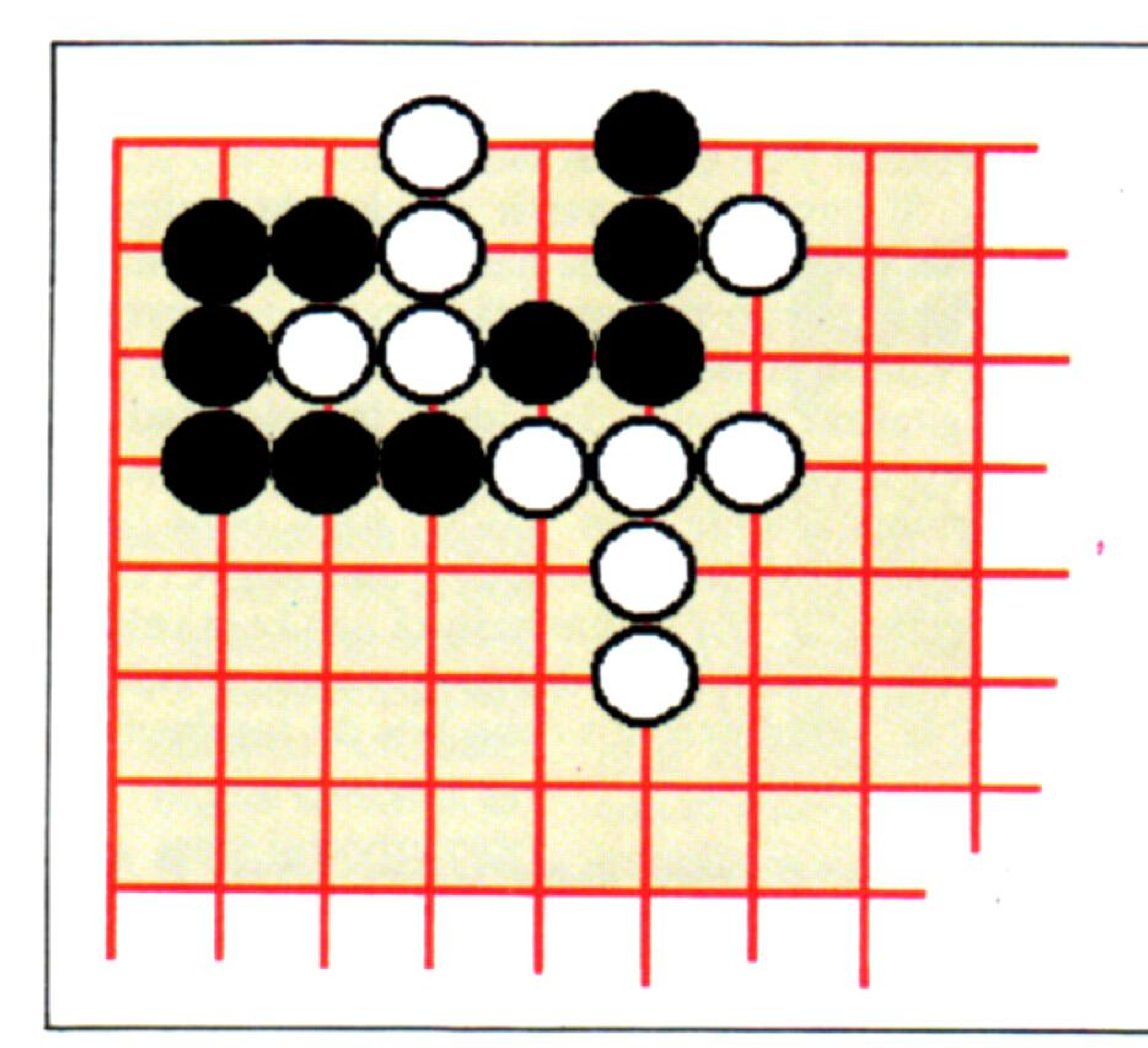
Técnicas de conversión A/D

El método de conversión que se suele emplear consiste en comparar una tensión conocida con otra desconocida (la que se mide). Los chips de comparación integrados (una clase de amplificadores operacionales, descritos anteriormente) resultan muy adecuados para esto, ya que pueden configurarse para dar una salida cuando (y sólo cuando) las dos tensiones de entrada son iguales. Un contactor binario se fija inicialmente a cero y cuenta, dando una tensión de salida (mediante una de las técnicas D/A descritas) que a la larga se hará igual a la tensión de entrada desconocida que se mide. Cuando la tensión desconocida y la tensión determinada por el convertidor D/A son iguales, el comparador produce una salida que detiene el contador binario y permite que se calcule la tensión. A la izquierda puede verse un circuito simplificado para este tipo de convertidor A/D.

Aunque existen muchos IC de un solo chip que pueden activar directamente visualizadores LCD o LED, diseñados específicamente para su aplicación a DVM, normalmente no resulta sencillo acoplarlos a los ordenadores, porque sus salidas sólo son adecuadas para atacar visualizadores digitales. Por tanto, hemos optado por basar nuestro diseño en el chip de medidor de panel digital ICL7135 de Analog Systems. Este chip proporciona salidas decimales codificadas en binario que pueden ser convertidas fácilmente en una forma adecuada para activar visualizadores de siete segmentos o ser acopladas a un ordenador.

El chip presenta una impedancia de entrada muy elevada y una entrada diferencial que permite medir tanto tensiones positivas como negativas. También permite una lectura de cero verdadero con una entrada de 0 V, las indicaciones de fuera margen mínimo y de margen máximo, y de polaridad, E/S lógica para su acoplamiento a un microprocesador y unos requisitos de reloj y de alimentación sencillos. Aunque es un chip caro, es mucho más barato que el circuito equivalente montado con componentes discretos, es más fácil de utilizar y ofrece muchas más posibilidades de funcionar bien a la puesta en marcha.

Para construir un tester de gran precisión sólo se necesitan dos cosas: resistencias de tolerancia restringida para el atenuador de entrada y el empleo de un DVM de precisión de algún amigo para fijar la tensión de referencia. No obstante, aun sin estos elementos, se podrán obtener resultados sumamente satisfactorios.



SEMEAI

Situación comprometida
Un método común para capturar fichas junto al margen del tablero consiste en asfixiarlas, táctica conocida como *semeai*. Es importante la cantidad de licencias que le quedan a cada grupo involucrado en una batalla *semeai*. Aquí, por ejemplo, las cosas parecen habérsele puesto muy difíciles al grupo de blancas en forma de L, dado que sólo tiene tres licencias, mientras que el correspondiente grupo de negras posee cuatro

En su lugar

Continuando con nuestro proyecto de programación de este juego oriental, centraremos nuestra atención en el módulo que le permitirá colocar fichas en el tablero

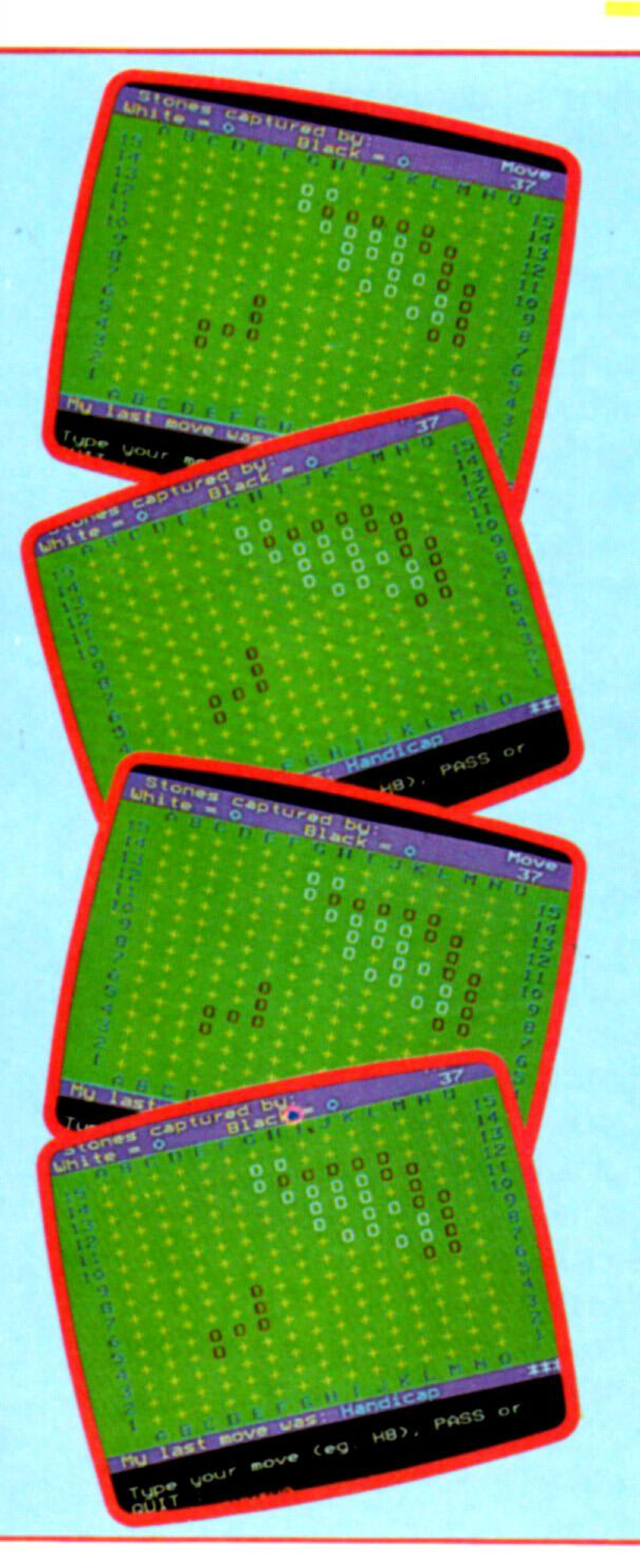
Debido a la forma estructurada y generalizada en la que se ha desarrollado el programa, es sumamente sencillo modificar el programa principal que llama a las rutinas (líneas 10-130) para que usted y algún amigo jueguen entre sí. Más adelante veremos cómo hacerlo, de modo que esté entonces en condiciones de utilizar el programa para partidas entre dos jugadores, con el ordenador actuando a modo de árbitro. La descripción del programa está referida a la versión para el BBC Micro; en las otras tres versiones se mantienen los números de línea y la estructura del programa.

La primera rutina que vamos a considerar es FNlegalidad (de la línea 3890 a la 4000). Esta función aceptará un movimiento, P%, realizado por el color C%, y devolverá un valor que indicará si el movimiento es legal o no. Esta comprobación asegura que:

1. La intersección (P%) esté vacante.
2. El jugador no capture una ficha en Ko.
3. La ficha del jugador no esté cometiendo suicidio.

Si examina la rutina que lee mensajes (de la línea 390 a la 560 en la versión para el BBC Micro), observará que estas tres condiciones coinciden con los mensajes dos, tres y cuatro (tres, cuatro y cinco en la versión para el Spectrum), respectivamente. Por consiguiente, FNlegalidad se ha diseñado para que devuelva alguno de estos números, o bien cero si el movimiento es legal.

El hecho de que la posición esté vacante se maneja mediante la comprobación de un cero en el byte apropiado del tablero, en la línea 3910. Para comprobar las otras dos posibilidades, se coloca



Módulo tres

BBC Micro:

```
1380 ko%=FALSE:fin%=FALSE
2310 :
2320 DEF PROCmovimiento__blancas
2330 LOCAL C%,L%,P%,X%,Y%,A$,C$,Y$
2340 atari2$="          "
2350 A$=fnentrada(21,8,4)
2360 IF A$="PASO" THEN ko%=FALSE:PROCimprimir__
     tablero:GOTO 2490
2370 IF A$="ABANDONO" THEN fin%=TRUE:ENDPROC
2380 X%=ASC(LEFT$(A$,1))-64
2390 Y$=MID$(A$,2)
2400 FOR C%=1 TO LEN(Y$)
2410    CS=MIDSS(Y$,C%,1)
2420    IF C$<"0" OR C$>"9" THEN
     PROCmensaje(23,1,A$):GOTO 2350
2430    NEXT
2440 Y%=VAL(Y$)
2450 IF X%<1 OR X%> 15 OR Y%<1 OR Y%> 15 THEN
     PROCmensaje(23,1,A$):GOTO 2350
2460 P%=16*Y%+X% : L%=FNlegalidad(P%,blancas%)
2470 IF L%>0 THEN PROCmensaje(23,L%,A$):GOTO 2350
2480 PROCefectuar__movimiento(P%,blancas%)
2490 PROCmensaje(23,0,"")
2500 ENDPROC
2510 :
2520 REM ************************
3620 :
3630 DEF PROCefectuar__movimiento(P%,C%)
3640 LOCAL A%,L%,N%
3650 tablero%? P%=C%
3660 FOR L%=1 TO 4 : A%=P%+dir%(L%) : IF
     tablero%?A%<> color%-C% THEN 3700
3670 PROCcontar(A%,color%-C%):IF clib%=0 THEN
     ko%=A%:N%=N%+cstn% :
     PROCsuprimir(A%,color%-C%)
3680    IF clib%=1 AND C%=negras% THEN
     atari1$="Atari":atari2$="
3690 IFclib%= 1 AND C%=blancas% THEN atari2$="Atari"
3700 NEXT
3710 IF N% <> 1 THEN ko%=FALSE
3720 captura%(C%)=captura%(C%)+N%
3730 PROCimprimir__tablero
3740 ENDPROC
3750 :
3760 REM ************************
3770 :
3780 DEF PROCsuprimir(P%,C%)
3790 IF tablero%?P%<>C% THEN ENDPROC
3800 tablero%?P%=0
3810 PROCsuprimir (P%+dir%(1),C%)
3820 PROCsuprimir (P%+dir%(2),C%)
3830 PROCsuprimir (P%+dir%(3),C%)
3840 PROCsuprimir (P%+dir%(4),C%)
3850 NEDPROC
3860 :
3870 REM ************************
3880 :
3890 DEF FNlegalidad(P%,C%)
3900 LOCAL A%,K%,L%,S%
3910 IF tablero%?P%<>0 THEN =2
3920 tablero%?P%=C%
3930 FOR L%=1 TO 4
3940    A%=P%+dir%(L%)
3950    IF tablero%?A%=color%-C% THEN
     PROCcontar(A%,color%-C%):IFclib%=0 THEN
     K%=K%+cstn%
3960    NEXT
3970 IF K%=0 THEN PROCcontar(P%,C%):IF clib%=0 THEN
     S%=4
3980 tablero%?P%=0
3990 IF P%=ko% AND K%=1 THEN  =3
4000 =S%
4010 :
4020 REM ************************
```

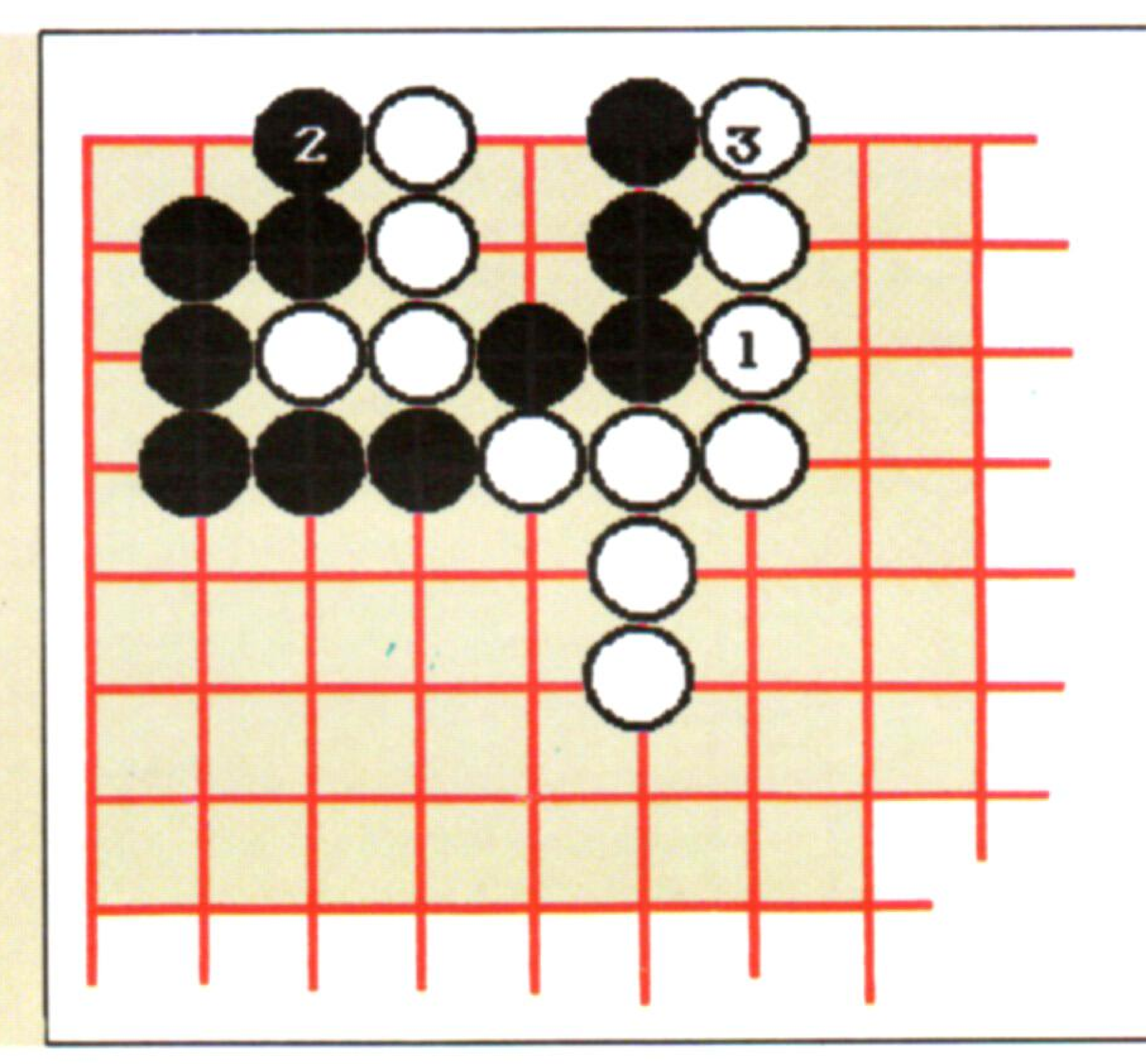


temporalmente en el tablero la ficha a jugar. El bucle, L%, cuenta luego las licencias de los grupos enemigos circundantes (utilizando PROCcontar, desarrollado en el último módulo). Dentro de este bucle, K% lleva el registro de la cantidad de fichas que serían muertas mediante el movimiento P%. Ahora se pueden efectuar las comprobaciones finales:

- Si no se mata ninguna ficha (K%=0) y la ficha a colocar no tiene licencias (clib%=0), la ficha debe estar cometiendo suicidio.
- Si se mata una ficha (K%=1) y la ficha a jugar se halla en la posición Ko (P%=ko%), se está violando la regla del Ko.

La comprobación del Ko quizá resulte algo difícil de comprender. Cuando un jugador captura una sola ficha, ko% se establece igual a la posición de la ficha capturada mediante la rutina efectuar_movimiento en las líneas 3670 y 3710. Si la nueva ficha a jugar está posicionada en este punto y está capturando sólo una ficha contraria, debe estar en el punto Ko y estar violando la regla del Ko. La ficha que esté capturando debe ser la última ficha jugada por el oponente.

La siguiente rutina es la que actualizará el tablero tras cada movimiento. PROCefectuar_movimiento comienza por colocar una ficha del color adecuado en el tablero, y un bucle comprueba el estado de cada uno de los grupos que la rodean. Si la nueva ficha llena la última licencia restante de un grupo contrario, ese grupo se suprimirá del tablero mediante una llamada a PROCsuprimir. Por el contrario, si sólo posee una licencia, entonces está en peligro de ser capturada al siguiente movimiento, de modo que se actualiza la serie atari adecuada. Recordará que atari es un aviso habitual, que se consigna cuando usted está a punto de capturar una ficha, o más, de su oponente. Por último, se actualiza la cantidad de fichas capturadas por el jugador

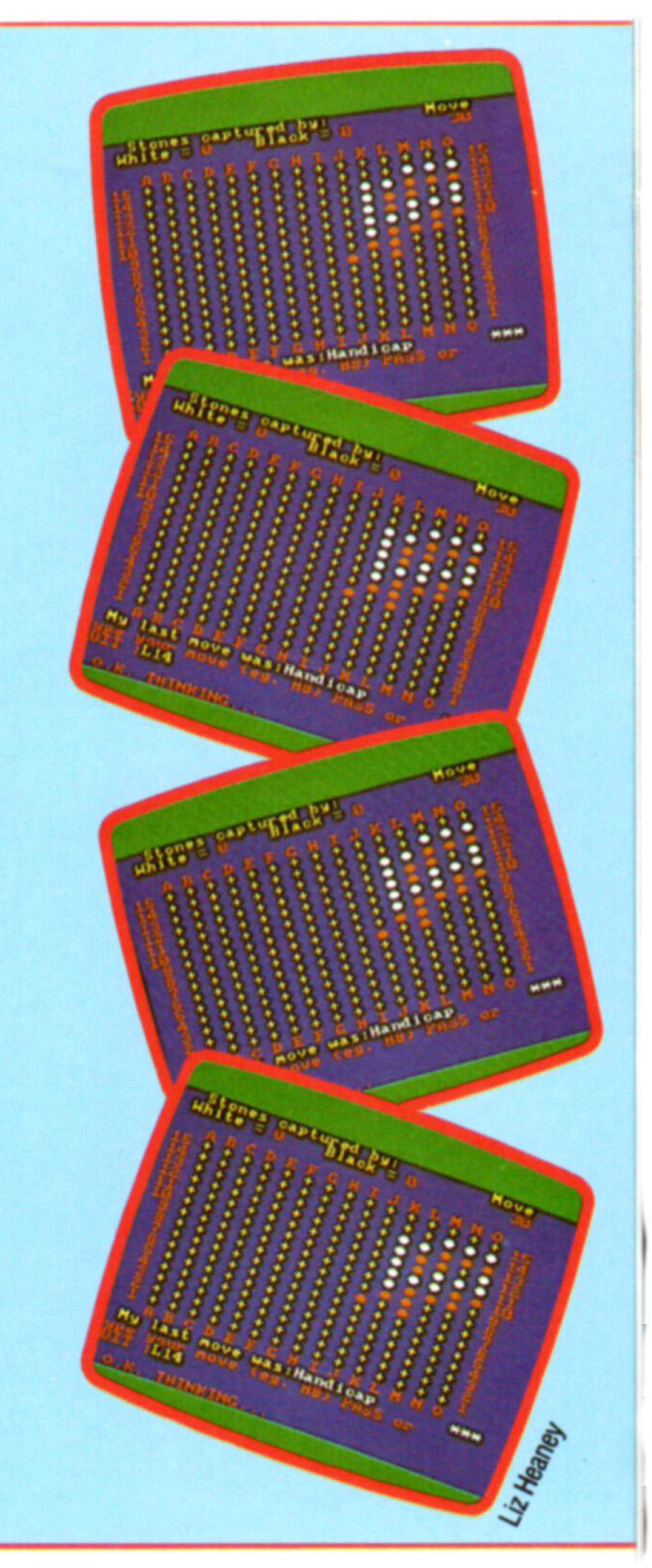


SEKI

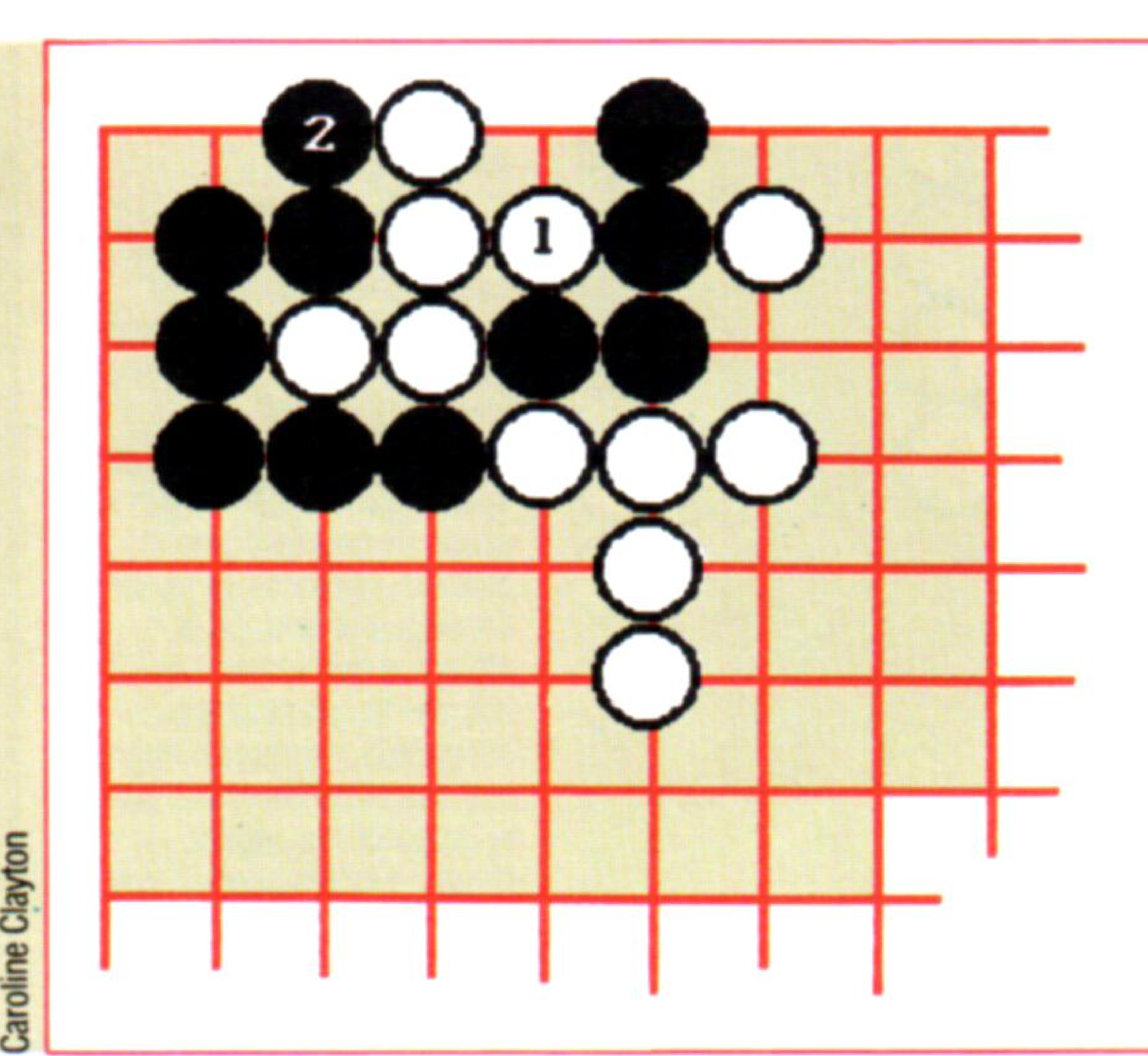
Situación neutra
Las batallas *semeai* casi siempre terminan ya sea con la captura del grupo más débil o bien en la situación de estancamiento denominada *seki*. Si las blancas atacan al grupo de negras desde el exterior, entonces se puede conseguir el *seki*. En la situación que vemos aquí, las blancas tienen su posición asegurada, porque comparten sus dos licencias restantes con el grupo de las negras. Ninguno de los bandos puede jugar en esta zona sin dejar a su grupo a merced de una captura

Amstrad CPC 464/664:

```
  60 MOVIMIENTO%=MOVIMIENTO%+1:GOSUB 2320
1380 KO%=0:FIN%=0
2310 :
2320 REM RUTINA MOVIMIENTO BLANCAS
2340 A2$="          "
2350 IP%=22:IM%=8:IW%=4:GOSUB 1990
2360 IF A$="PASO" THEN KO%=0:GOSUB 1730:GOTO
     2490
2370 IF A$="ABANDONO" THEN FIN%=1:RETURN
2380 X%=ASC(LEFT$(A$,1))-64
2390 Y$=MID$(A$,2)
2400 FOR WC=1 TO LEN(Y$)
2410 C$=MID$(Y$,WC,1)
2420 IF C$<"0" OR C$>"9" THEN
     MP%=24:MM%=1:O$=A$:GOSUB 2160:GOTO 2350
2430 NEXT
2440 Y%=VAL(Y$)
2450 IF X%>0 AND X%<16 AND Y%>0 AND Y%<16 GOTO
     2460
2455 MP%=24:MM%=1:O$=A$:GOSUB 2160:GOTO 2350
2460 WP%=16*Y%+X%:LP%=WP%:LC%=BLANCAS
     %:GOSUB 3890
2470 IF LL%>0 THEN MP%=24:MM%=LL%:O$=A$:GOSUB
     2160:GOTO 2350
2480 MP%=WP%:MC%=BLANCAS%:GOSUB 3630
2490 MP%=24:MM%=0:O$="":GOSUB 2160
2500 RETURN
2510 :
2520 REM ***********************
3620 :
3630 REM RUTINA EFECTUAR-MOVIMIENTO
3640 N%=0
3650 POKE TABLERO+MP%,MC%
3660 FOR K=1 TO 4:A=MP%+DIR(K):IF
     PEEK(TABLERO+A)<>COLOR%-MC% GOTO 3700
3670 CP%=A:CC%=COLOR%-MC%:GOSUB 4040:IF
     CLIB%<>0 GOTO 3680
3675 KO%=A:N%=N%+CSTN%:RP%=A:RC%=COLOR%-
     MC%:GOSUB 3780
3680 IF CLIB%=1 AND MC%=NEGRAS% THEN
     A1$="ATARI":A2$="          "
3690 IF CLIB%=1 AND MC%=BLANCAS% THEN A2$=
     "ATARI"
3700 NEXT
3710 IF N%<>1 THEN KO%=0
3720 CAPTURA%(MC)=CAPTURA%(M)+N%
3730 GOSUB 1730
3740 RETURN
3750 :
3760 REM ***********************
3770 :
3780 REM RUTINA SUPRIMIR
3790 IF PEEK(TABLERO+RP%)<>RC% THEN
     RETURN
3800 POKE TABLERO+RP%,0
3805 SK%(PILA%)=RP%:PILA%=PILA%+1
3810 RP%=SK%(PILA%-1)+DIR%(1):GOSUB 3780
3820 RP%=SK%(PILA%-1)+DIR%(2):GOSUB 3780
3830 RP%=SK%(PILA%-1)+DIR%(3):GOSUB 3780
3840 RP%=SK%(PILA%-1)+DIR%(4):GOSUB 3780
3845 PILA%=PILA%-1:RP%=SK%(PILA%)
3850 RETURN
3860 :
3870 REM ***********************
3880 :
3890 REM RUTINA LEGALIDAD
3900 LL%=0:LK%=0
3910 IF PEEK(TABLERO+LP%)<>0 THEN LL%=2:
     RETURN
3920 POKE TABLERO+LP%,LC%
3930 FOR K=1 TO 4
3940 LA%=LP%+DIR%(K)
3950 IF PEEL(TABLERO+LA%)<>COLOR%-LC% GOTO
     3960
3955 CP%=LA%:CC%=COLOR%-LC%:GOSUB 4040:IF
     CLIB%=0 THEN LK%=LK%+CSTN%
3960 NEXT
3970 IF LK%=0 THEN CP%=LP%:CC%=LC%:GOSUB 4040:IF
     CLIB%=0 THEN LL%=4
3980 POKE TABLERO+LP%,0
3990 IF LP%=KO% AND LK%=1 THEN LL%=3
4000 RETURN
4010 :
4020 REM ***********************
```



Liz Heaney



Situación límite
Si las blancas cometen el error de jugar en estas licencias internas, como vemos aquí, quedarán en situación de ser capturadas. Después de que las negras juegan la ficha 2, al grupo de las blancas sólo le queda una licencia y la batalla, efectivamente, está perdida

Caroline Clayton

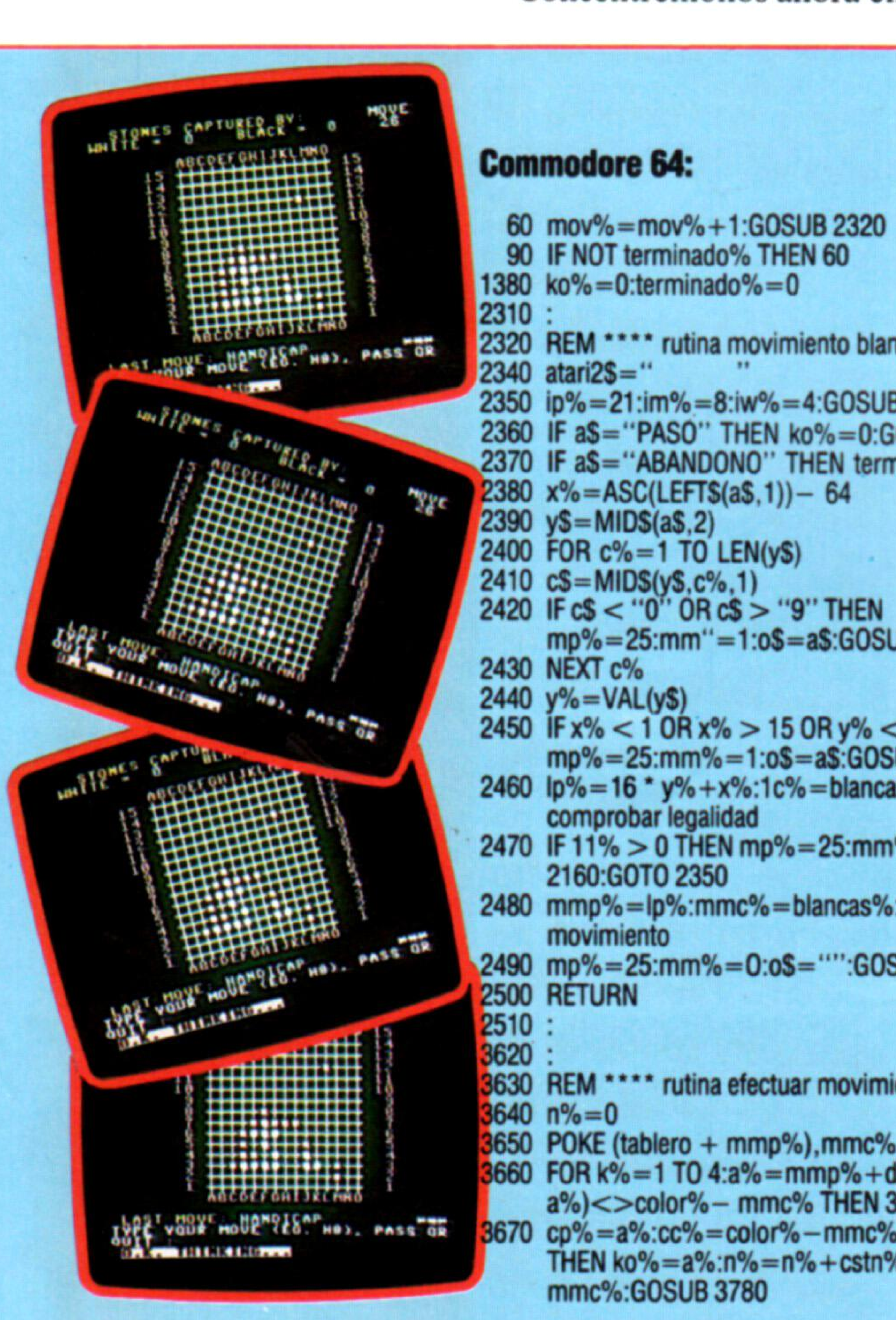
actual y se reimprime la pantalla utilizando la rutina imprimir__tablero.

Cuando se captura una ficha o un grupo de fichas, se llama PROCsuprimir para eliminarlas del tablero. Esta rutina trabaja de una forma repetitiva similar a PROCbuscar. Inicialmente suprime del tablero la ficha de la posición P%. Luego se llama repetidamente a sí misma para eliminar todas las fichas del mismo color que hubiera inmediatamente al norte, este, sur u oeste de la posición actual. Nuevamente, en las versiones para el Commodore 64, el Spectrum y el Amstrad se utiliza la «pila» para simular la repetición.

Concentrémonos ahora en la rutina que le permitirá entrar y realizar movimientos: PROCmovimiento__blancas. Puesto que ya contamos con algunas útiles rutinas a las que podemos llamar, la mayor parte del código tiende a comprobar la validez de la entrada. La rutina FNentrada (línea 2350) le permitirá entrar cuatro caracteres cualesquiera, de modo que tendrá que comprobar si usted ha entrado PASO, ABANDONO o una posición legal del tablero.

PASO sólo requiere restablecer la bandera Ko antes de retornar, mientras que ABANDONO establece en TRUE el valor lógico de fin del juego, de modo que al retornar el programa sale del bucle principal del movimiento, de la línea 60 a la 90. Si la entrada no es PASO ni ABANDONO, el código del programa divide la entrada en coordenadas horizontales (de A a O) y verticales (de 1 a 15), que se combinan en una posición del tablero, P%. Observará que aquí no podemos utilizar con facilidad la función VAL. Ello se debe a que la entrada puede ser de dos o tres caracteres de longitud y VAL(MIDS(«A1?»,2)) nos daría el resultado legal de uno, aunque al final de la serie haya un signo de interrogación adulterado. Si la entrada es válida y FNlegalidad confirma que el movimiento es legal, la rutina llama a PROCefectuar__movimiento, restablece la zona de mensajes llamando a PROCmensaje con el mensaje O.K.ESTOY PENSANDO (listo para que juegue el ordenador) y luego termina.

A lo largo de este programa hemos utilizado las constantes negras% y blancas% en lugar de los verdaderos números uno y dos. La única excepción se produce en PROCimprimir__tablero, donde los elementos de la matriz captura% muestran la cantidad



Commodore 64:

```
  60 mov%=mov%+1:GOSUB 2320
  90 IF NOT terminado% THEN 60
1380 ko%=0:terminado%=0
2310 :
2320 REM **** rutina movimiento blancas ****
2340 atari2$="        "
2350 ip%=21:im%=8:iw%=4:GOSUB 1990:REM entrada
2360 IF a$="PASO" THEN ko%=0:GOSUB 1730:GOTO 2490
2370 IF a$="ABANDONO" THEN terminado%=1:RETURN
2380 x%=ASC(LEFT$(a$,1))- 64
2390 y$=MID$(a$,2)
2400 FOR c%=1 TO LEN(y$)
2410 c$=MID$(y$,c%,1)
2420 IF c$ < "0" OR c$ > "9" THEN
     mp%=25:mm"=1:o$=a$:GOSUB 2160:GOTO 2350
2430 NEXT c%
2440 y%=VAL(y$)
2450 IF x% < 1 OR x% > 15 OR y% < 1 OR y% > 15 THEN
     mp%=25:mm%=1:o$=a$:GOSUB 2160:GOTO 2350
2460 lp%=16 * y%+x%:1c%=blancas%:GOSUB 3890:REM
     comprobar legalidad
2470 IF 11% > 0 THEN mp%=25:mm%=11%:o$=a$:GOSUB
     2160:GOTO 2350
2480 mmp%=lp%:mmc%=blancas%:GOSUB 3630:REM efectuar
     movimiento
2490 mp%=25:mm%=0:o$="":GOSUB 2160
2500 RETURN
2510 :
3620 :
3630 REM **** rutina efectuar movimiento ****
3640 n%=0
3650 POKE (tablero + mmp%),mmc%
3660 FOR k%=1 TO 4:a%=mmp%+dir%(k%):IF PEEK(tablero +
     a%)<>color%- mmc% THEN 3700
3670 cp%=a%:cc%=color%-mmc%:GOSUB 4040:IF clib%=0
     THEN ko%=a%:n%=n%+cstn%:rp%=a%:rc%=color%-
     mmc%:GOSUB 3780
3680 IF clib%=1 AND mmc%=blancas% THEN atari2$="Atari"
3700 NEXT k%
3710 IF n% <> 1 THEN ko%=0
3720 captura%(mmc%)=captura%(mmc%)+n%
3730 GOSUB 1730:REM imprimir tablero
3740 RETURN
3750 :
3760 REM ************************
3770 :
3780 REM rutina suprimir
3790 IF PEEK (tablero+rp%)<>rc% THEN RETURN
3800 POKE(tablero+rp%),0
3805 s(pila%)=rp%:pila%=pila%+1
3810 rp%=s(pila%-1)+dir%(1):GOSUB 3780
3820 rp%=s(pila%-1)+dir%(2):GOSUB 3780
3830 rp%=s(pila%-1)+dir%(3):GOSUB 3780
3840 rp%=s(pila%-1)+dir%(4):GOSUB 3780
3845 pila%=pila%-1:rp%=s(pila%)
3850 RETURN
3860 :
3870 REM ************************
3880 :
3890 REM rutina legalidad
3900 11%=0:1k%=0
3910 IF PEEK(tablero+1p%)<> 0 THEN 11%=2:RETURN
3920 POKE(tablero+1p%),lc%
3930 FOR k%=1 TO 4
3940 la%=lp%+dir%(k%)
3950 IF PEEK(tablero+la%)=color%-lc% THEN
     cp%=la%:cc%=color%-lc%:GOSUB 4040:IF clib%=0
     THEN lk%=lk%+cstn%
3960 NEXT k%
3970 IF lk%=0 THEN cp%=lp%:cc%=lc%:GOSUB 4040:IF
     clib%=0 THEN 11%=4
3980 POKE (tablero+lp%),0
3990 IF lp%=ko% AND lk%=1 THEN 11%=3
4000 RETURN
4010 :
4020 REM ************************
```



de fichas capturadas por cada bando. Una ventaja de este método es que nos permite modificar fácilmente la forma en la que representan los colores los bytes del tablero.

Ahora podemos utilizar esta facilidad para hacer un juego de dos jugadores, cambiando las siguientes líneas (véase *Complementos al* BASIC para las versiones destinadas a otras máquinas):

```
60 movimiento%=movimiento%+1:negras%=
   1:blancas%=2:PROCmovimiento__blancas%
80 movimiento%=movimiento%+1:negras%=
   2:blancas%=1:PROCmovimiento__blancas
```

Con ello tan sólo se cambian los valores reales de negras% y blancas% llamando cada vez a PROCmovimiento__blancas. Puesto que todas las rutinas se han diseñado para permitir cualquiera de los valores, el ordenador permitirá que negras y blancas jueguen alternativamente, utilizando ambas PROCmovimiento__blancas para comprobar la legalidad de cada movimiento y visualizarlos en el tablero. En el próximo módulo tendremos que cambiar estas líneas para que pueda jugar el ordenador. De momento, esta ínfima modificación le permitirá organizar partidas entre dos jugadores.

Complementos al BASIC

Insertando estas líneas en los listados de programa que ofrecemos, podrá disponer de un juego para dos jugadores, con el ordenador actuando como un tablero inteligente, comprobando los movimientos legales y llevando los marcadores.

Commodore 64

```
60 MOVIMIENTO%=MOVIMIENTO%+:NEGRAS%
   = 1:BLANCAS%=2:GOSUB 2320
80 MOVIMIENTO%=MOVIMIENTO%+1:NEGRAS%
   =2:BLANCAS%=1:GOSUB 2320
```

Spectrum

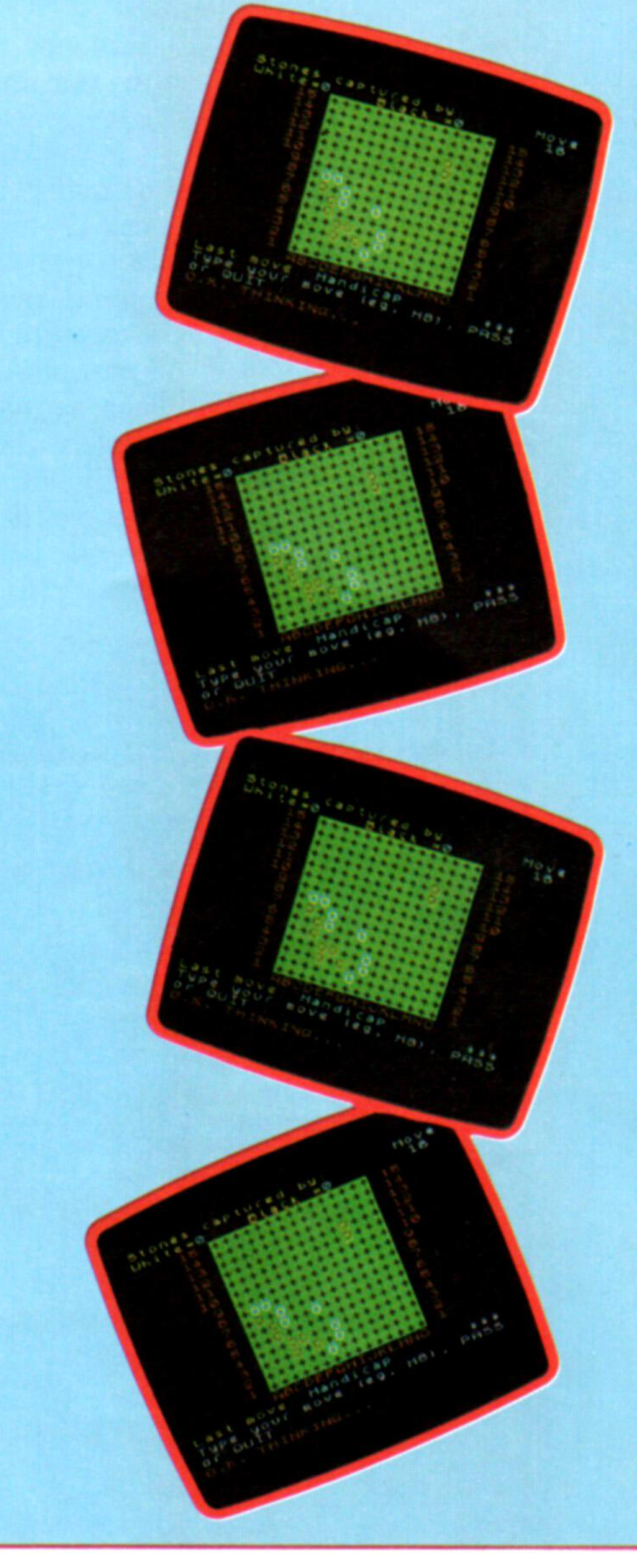
```
60 LET movimiento=movimiento+1:LET negras=
   1:LET blancas=2:GO SUB 2320
80 LET movimiento=movimiento+1:LET negras=
   2:LET blancas=1:GO SUB 2320
```

Amstrad CPC 464/664

```
60 mov%=mov%+1:negras%=
   1:blancas%=2:GOSUB 2320
80 mov%=mov%+1:negras%=
   2:blancas%=1:GOSUB 2320
```

Sinclair Spectrum:

```
  60 LET movimiento=movimiento+1:GO SUB
     2320
1380 LET ko=0: LET fin=0
2310 :
2320 REM rutina movimiento blancas
2340 LET y$="          "
2350 LET ip=19: LET im=9: LET iw=4: GO SUB
     1990
2360 IF a$="PASO" THEN LET ko=0: GO SUB
     1730: GO TO 2490
2370 IF a$="ABANDONO" THEN LET
     fin=1:RETURN
2380 LET x=CODE (a$(1))-64
2390 LET z$=a$(2 TO)
2400 FOR c=1 TO LEN z$
2410 LET c$=z$(c)
2420 IF c$ < "0" OR c$ > "9" THEN LET mp=21:
     LET mm=2: LET o$=a$:GO SUB 2160:GO TO
     2350
2430 NEXT c
2440 LET y=VAL z$
2450 IF x<1 OR x>15 OR y<1 OR y>15 THEN LET
     mp=21: LET mm=2: LET o$=a$: GO SUB
     2160:GO TO 2350
2460 LET lp=16*y+x: LET lc=blancas: GO SUB
     3890
2470 IF 11 > 0 THEN LET mp=21: LET mm=11:
     LET o$=a$: GO SUB 2160: GO TO 2350
2480 LET mmp=lp: LET mmc=blancas: GO SUB
     3630
2490 LET mp=21: LET mm=1: LET o$="": GO
     SUB 2160
2500 RETURN
2510 :
2520 REM ***********************
3620 :
3630 REM rutina efectuar movimiento
3640 LET n=0
3650 POKE tablero+mmp,mmc
3660 FOR k=1 TO 4:LET a=mmp+d(k): IF PEEK
     (tablero+a)<>color-mmc THEN GO TO 3700
3670 LET cp=a: LET cc=color-mmc: GO SUB 4040:
     IF clib=0 THEN LET ko=a: LET n=n+cstn:
     LET rp=a: LET rc=color-mmc: GO SUB 3780
3680 IF clib=1 AND mmc=negras THEN LET
     x$="Atari": LET y$="          "
3690 IF clib=1 AND mmc=blancas THEN LET
     y$="Atari"
3700 NEXT k
3710 IF n<>1 THEN LET ko=0
3720 LET c(mmc)=c(mmc)+n
3730 GO SUB 1730
3740 RETURN
3750 :
3760 REM ***********************
3770 :
3780 REM rutina suprimir
3790 IF PEEK(tablero+rp)<>rc THEN
     RETURN
3800 POKE tablero+rp,0
3805 LET s(pila)=rp: LET pila=pila+1
3810 LET rp=s(pila-1)+d(1):GO SUB 3780
3820 LET rp=s(pila-1)+d(2):GO SUB 3780
3830 LET rp=s(pila-1)+d(3):GO SUB 3780
3840 LET rp=s(pila-1)+d(4):GO SUB 3780
3845 LET pila=pila-1: LET rp=s(pila)
3850 RETURN
3860 :
3870 REM ***********************
3880 :
3890 REM rutina legalidad
3900 LET 11=0: LET lk=0
3910 IF PEEK (tablero+lp)<>0 THEN LET
     ll=3:RETURN
3920 POKE tablero+lp,lc
3930 FOR k=l TO 4
3940 LET la=lp+d(k)
3950 IF PEEK (tablero+la)=color-lc THEN LET
     cp=la: LET cc=color-lc: GO SUB 4040 : IF
     clib=0 THEN LET lk=lk+cstn
3960 NEXT k
3970 IF lk=0 THEN LET cp=lp: LET cc=lc: GO SUB
     4040: IF clib=0 THEN LET ll=5
3980 POLE tablero+lp,0
3990 IF lp=ko AND lk=1 THEN LET ll=4
4000 RETURN
4010 :
4020 REM ***********************
```



Enterados

La Interface 1 del Spectrum también puede servir para acceder a una red de área local. Veamos cómo

Ya hemos analizado el modo cómo un programador en lenguaje máquina puede emplear las facilidades de la Interface 1 para controlar microdrives o acceder a la puerta serial RS232. En lo que llevamos dicho, se han visto ya los principios de trabajo en red así como ofrecimos un sencillo juego de red para el Spectrum. Nos ceñiremos ahora al estudio de las facilidades disponibles para un programador en lenguaje máquina. En nuestro análisis emplearemos el siguiente convenio en la denominación de las máquinas concernientes a una red: SELF representa la máquina de usted, IRIS es otra máquina distinta de la red con la que usted desea comunicar.

Transmisión por red

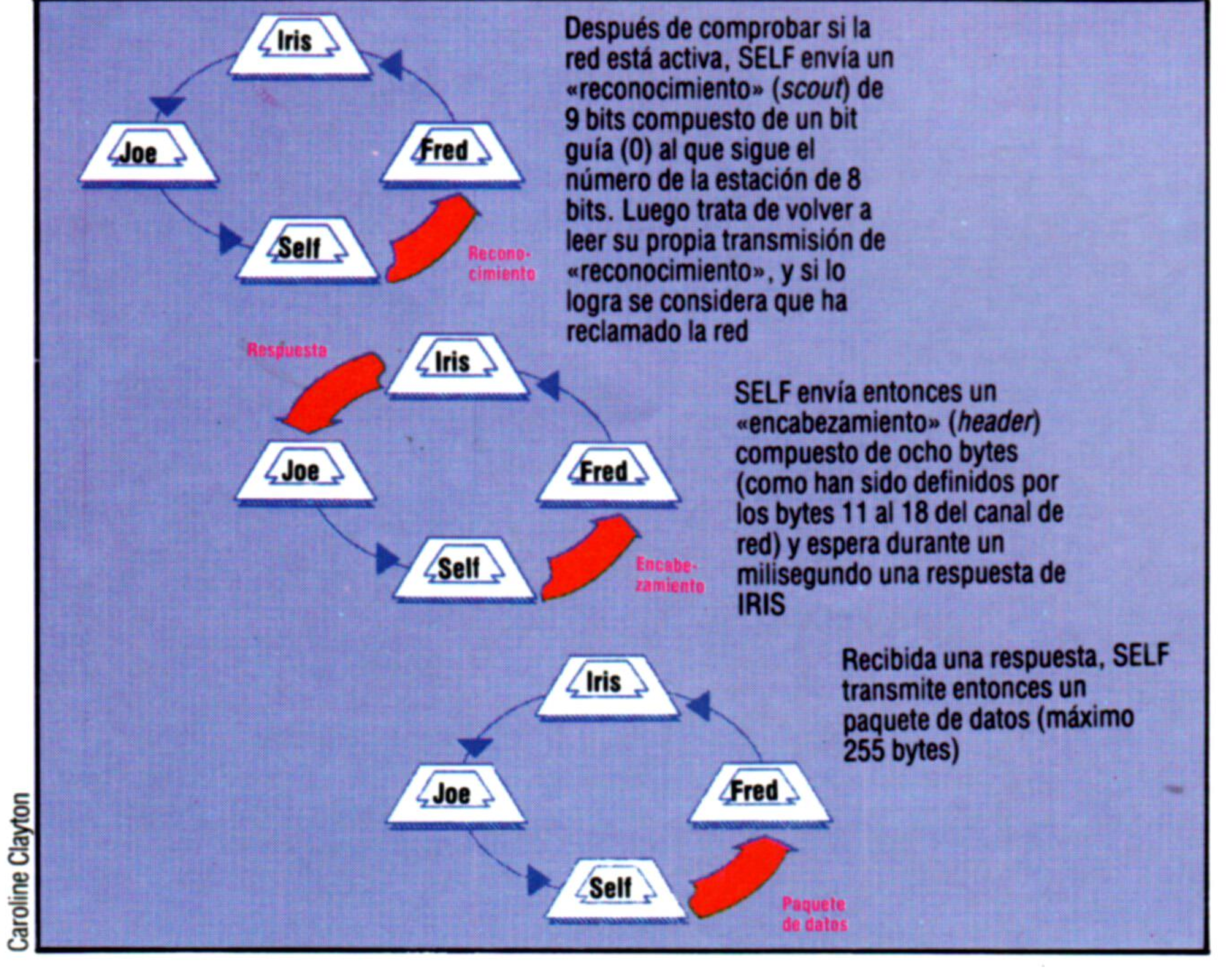
Hay dos maneras de transmitir programas y datos por una red. La primera es la *difusión* (*broadcast*) por las ondas. Toda máquina conectada a la red la puede «oír» si está «sintonizada» cuando se realiza la emisión. Se trata de un modo muy eficaz de distribuir un programa a varias máquinas con gran rapidez. La segunda manera es la *transmisión dirigida* (*directed transmission*), en la que SELF especifica el IRRIS que ha de recibir la información transmitida. Si el IRIS especificado no está a la escucha, la máquina SELF intentará reiteradamente establecer la comunicación.

El método por el cual las máquinas se comunican a través de la red es, sin duda, curioso. Una máquina que desea transmitir datos lo primero que hace es esperar hasta que la red guarde silencio, o «descanse». Para alejar el peligro de competencias en la red, lo que produciría una «confusión» de datos, cada máquina espera durante un intervalo de tiempo aleatorio después de haber comprobado que la red está libre y antes de intentar reclamarla para su uso. Si la red está todavía activa, esperará de nuevo.

Una vez que la red se encuentra libre, la máquina SELF que está transmitiendo los datos envía un *reconocimiento* (*scout*) sobre la red. Se trata del número de la estación de la máquina transmitido de tal modo que puede ser leído en retorno por SELF en la misma red. Cuando esto se cumple, se dice que SELF ha *reclamado* la red.

Ya puede enviarse un *paquete* de datos. Éste se compone de un encabezamiento, que contiene datos sobre el bloque de datos que sigue y un bloque de datos de 255 bytes. Lo que sucede después es función del tipo de transmisión. Si es una transmisión dirigida, SELF espera durante un milisegundo una señal de respuesta desde el IRIS deseado. Si no se recibe tal señal, tras un retardo de ocho milisegundos, SELF envía el reconocimiento y el encabezamiento de nuevo. No envía el bloque de datos hasta recibir la señal de respuesta. Si la transmisión es un difusión, no hay que esperar respuesta y el bloque de datos se envía sin más a la red.

De emisora a emisora
Una operación sobre red ante todo exige «reclamar» la red, a lo que seguirá la transmisión de los datos (iniciada con la necesaria información de encabezamiento). Este esquema muestra el flujo de datos en una red de cuatro estaciones, donde la estación SELF realiza una transmisión «dirigida» a la estación IRIS



Caroline Clayton

Códigos de enganche y variables de sistema

Consideremos ahora que se está utilizando la red desde el lenguaje máquina. Cuatro son los códigos de enganche relativos y dos importantes variables de sistema de la Interface 1:

- NTSTAT: Se encuentra en la dirección 23749 y se establece con el número de estación de la máquina. La instrucción FORMAT, empleada con la red, establece la variable de sistema. El valor por defecto es uno.
- DSTR1: Ya hemos encontrado esta variable al examinar los microdrives. Está en las direcciones 23766 y 23767 y se establece para que contenga el número de estación del IRIS con la que SELF desea comunicar. Poniéndola a cero, dará una transmisión por difusión de los datos.

Respecto al empleo actual de los códigos de enganche, volvemos a un sistema similar al usado con los microdrives, donde hay que abrir un canal de red antes de hacer otra cosa. Un canal de red se establece con el código de enganche 45 (en el dibujo mostramos su estructura). El canal tiene una longitud de 276 bytes, y se inserta en el «área de datos de canal» de la memoria como cualquier otro canal.

El encabezamiento enviado a la red se compone

de todos los bytes que van desde NCIRIS a NCHCS inclusive. El bloque de datos enviados por la red se compone de los bytes 21 al 275 del espacio de canal. Los bytes 19 y 20 de los datos de canal son sólo relevantes cuando el canal está siendo usado para leer datos desde la red. Como para el sistema de canal del microdrive, la escritura y lectura del canal puede hacerse fácilmente dando su dirección a CURCHL y después empleando RST #10 y CALL #15E6 (como con los microdrives). Sin embargo, aquí los datos y el encabezamiento se ponen en la red cuando queremos escribir el byte número 256 en el área de datos de canal. Veamos ahora los códigos de enganche:

- *Código de enganche 45*: Abre un canal de red. Para usarlo, se establece NTSTAT con el número de estación de SELF, y DSTR1 se establece con el número de estación de IRIS. Esto sirve cuando el canal ha sido abierto para lectura o escritura; pero si está leyendo, es evidente que IRIS transmitirá los datos a SELF.

Nuevamente hay que asegurarse de que existen las variables de sistema de la Interface 1. Entonces el código de enganche puede ser usado de la manera habitual, y al retorno el registro IX contiene la dirección de inicio del canal que se ha establecido.

Como ya se ha mencionado, los bytes pueden ser enviados a lo largo de la red por medio de RST =10 y ser recibidos por CALL #15E6. Sin embargo, cuando son empleados con la red, el registro IX puede alterarse. Pero como el contenido de este registro, o al menos la dirección del canal, se necesitará para cerrar el canal una vez hayamos terminado con él, interesa conservar el contenido de IX antes de usar estas rutinas.

- *Código de enganche 46*: Cierra el canal de red, y es llamado con la dirección de inicio del canal tomado del registro IX. Si el canal ha sido usado para escritura, cualquier dato que todavía esté en el área de datos de canal queda escrito en la red. Una vez cerrado un canal, la memoria entre el final del canal y STKEND se desplaza hacia abajo.

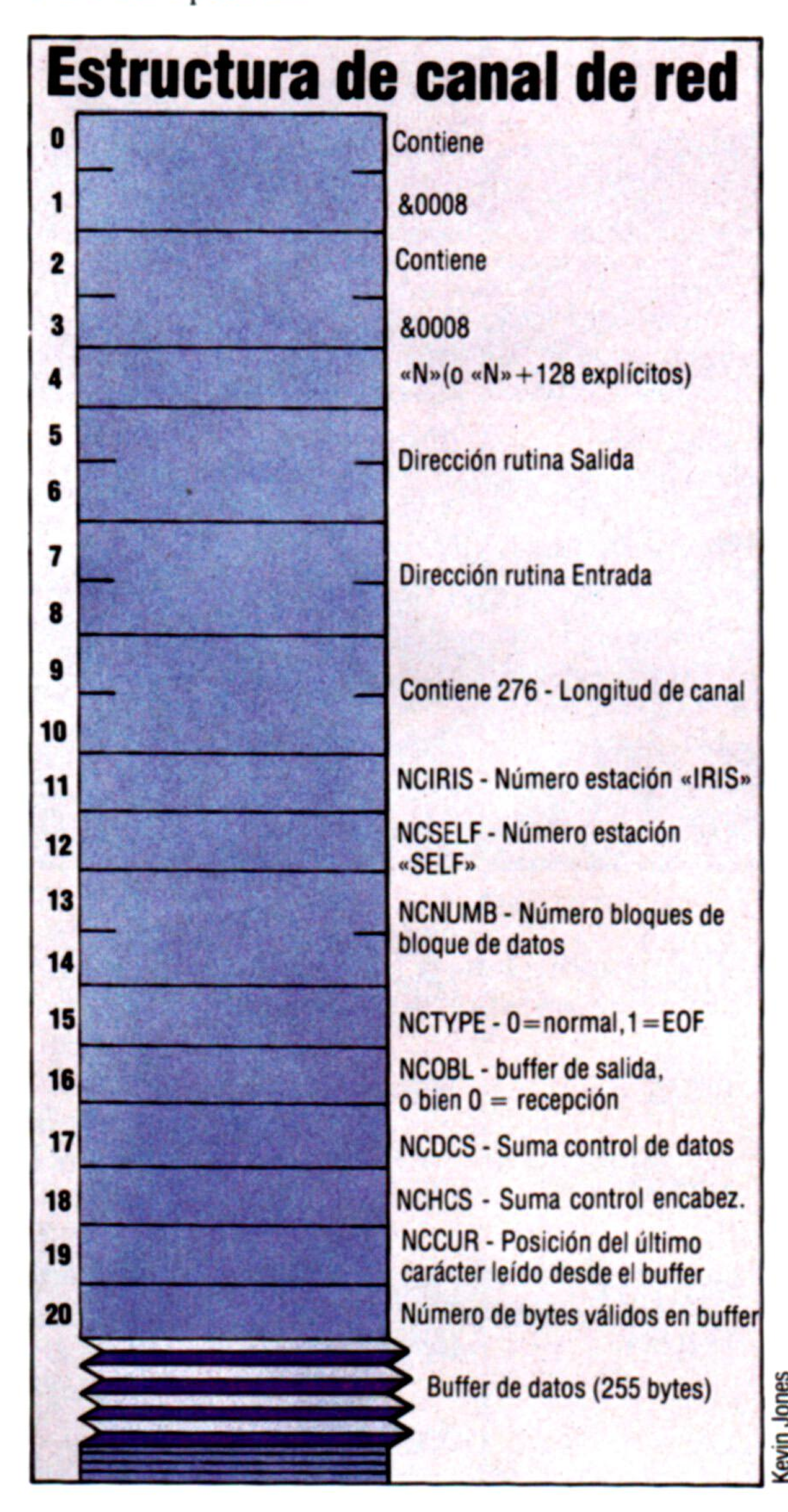
- *Código de enganche 47*: Permite la lectura de un paquete específico desde la red. La variable de canal NCNUMB tendrá el número adecuado de bloque: se supone que NCSELF y NCIRIS contendrán los valores apropiados (generalmente se establecen en el acto de abrir un canal). Tras el empleo del código de enganche, se hará un ensayo de recepción del paquete. Al retorno, el flag C se pondrá a uno si ocurre un final de tiempo, o si los datos estuvieren alterados de alguna manera y no concuerdan las sumas de comprobación. Así, inspeccionando el estado del flag de arrastre (*carry*) puede pedir repetidas veces un mismo paquete de la red. Si todo va bien, y el paquete es recibido, entonces se envía un asentimiento si es necesario y se incrementa NCNUMB.

- *Código de enganche 48*: Este código transmite un paquete de datos a través de la red, y por eso ha de considerarse como la rutina principal de la transmisión de datos. Se emplea con un canal abierto, cuya dirección se pasa a la rutina desde el registro IX. El registro A se activa para contener el tipo de archivo del bloque de datos que está siendo enviado desde el canal. Se pone a uno cuando el bloque que se está enviando es un bloque de datos *end of file* (fin de archivo); de lo contrario, para el bloque de datos normal, estará a cero. Los datos son enviados como difusión o como transmisión dirigida, según el valor contenido en DSTR1 cuando es abierto el canal. En el «extremo receptor», si se aceptan los datos, se leerá el encabezamiento en el área de encabezamiento del canal y quedarán disponibles en NCOBL el número de bytes enviados, indicando NCTYPE si el paquete recibido es o no un paquete *end of file*.

Si se consigue una transmisión acertada, la variable de canal NCNUMB añadirá una unidad a su valor.

Cuando se emplea la red LAN (*local area network*) se puede, pues, escoger entre dos maneras diferentes de enviar y recibir datos según sea la aplicación. Si está en una red pequeña, quizá con una sola máquina distinta de la suya, el método disponible a través de los códigos de enganche 48 y 47 puede quedar algo «saturado», por lo que el sencillo empleo de RST #10 y CALL #15E6 bastará.

Con esto completamos nuestro examen de la Interface 1 en su capacidad para comunicarse con otros ordenadores o periféricos. En un próximo capítulo estudiaremos el empleo de la Interface 1 para aumentar el número de las instrucciones del BASIC del Spectrum.



Estructura de canal de red
El código de enganche 45 establece un canal de red de 276 bytes, de los cuales 20 bytes se emplean como información de encabezamiento y 255 bytes son un buffer de los datos para ser transmitidos. Este esquema muestra la disposición de los bytes en el encabezamiento del canal

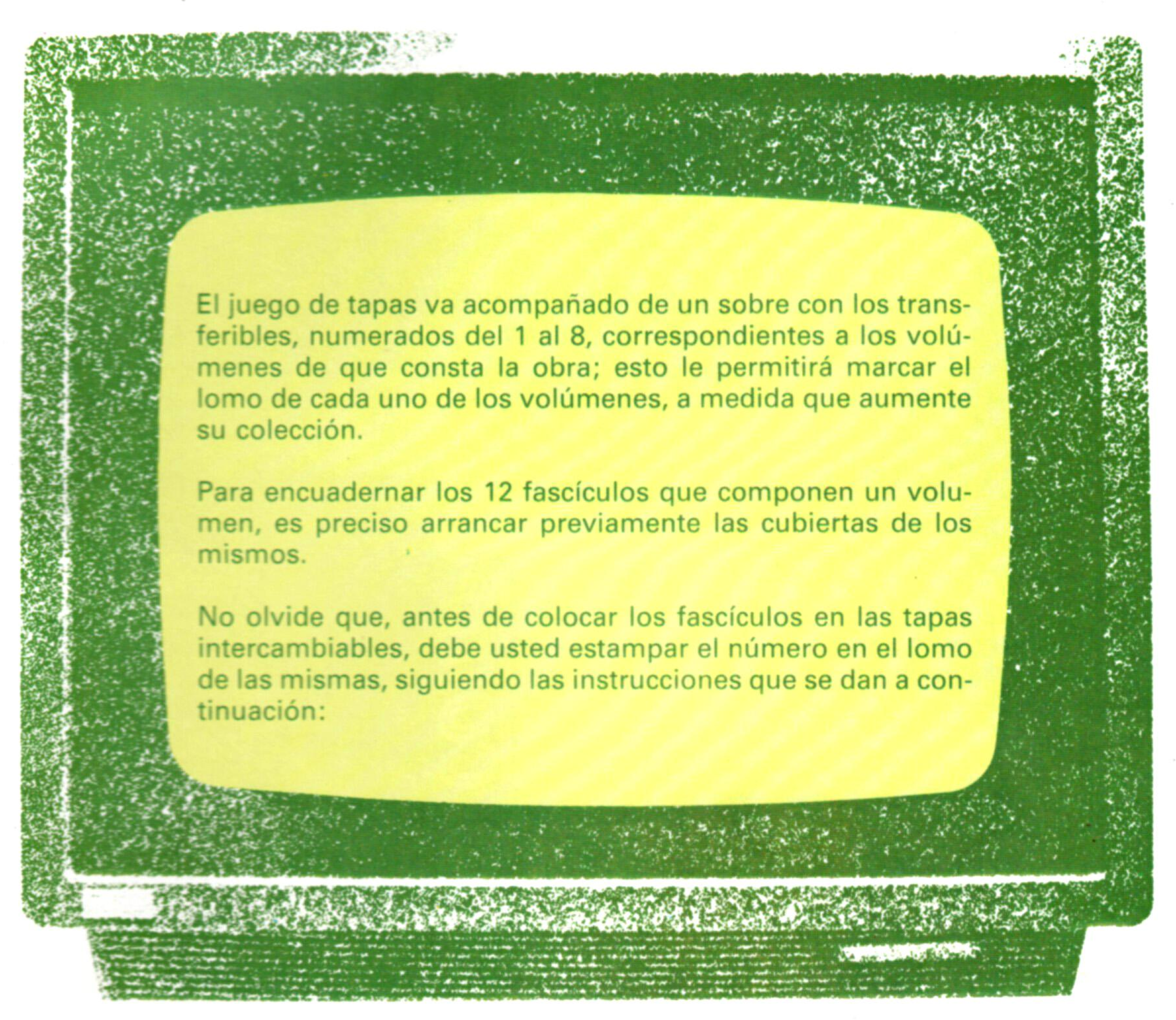
Datos básicos (IV)

Por cortesía de la Commodore Business Machines, reproducimos otro fragmento del mapa de memoria del Commodore 64

ETIQUETA	DIRECCIÓN HEXA	POSICIÓN DECIMAL	DESCRIPCIÓN
DFLTN	0099	153	Dispositivo entrada por defecto (0)
DFLTO	009A	154	Dispositivo salida (CMD) por defecto (3)
PRTY	009B	155	Paridad carácter cinta
DPSW	009C	156	Flag: byte de cinta recibido
MSGFLG	009D	157	Flag: $80 = Modo directo, $00 = Programa
PTR1	009E	158	Log error paso 1 de cinta
PTR2	009F	159	Log error paso 2 de cinta
TIME	00A0-00A2	160-162	Reloj instantáneo tiempo real 1/60 seg (aprox.)
	00A3-00A4	163-164	Área datos temporales
CNTDN	00A5	165	Contador sincr. de cassette
BUFPNT	00A6	166	Puntero: buffer E/S cinta
INBIT	00A7	167	Bits entrada RS-232 / Temp. cassette
BITCI	00A8	168	Contador bits entrada RS-232/Temp. cassette
RINONE	00A9	169	Flag RS-232: comprobación bit inicio
RIDATA	00AA	170	Byte entrada RS-232 buffer/temp. cassette
RIPRTY	00AB	171	Paridad entrada RS-232/contador corto cassette
SAL	00AC-00AD	172-173	Puntero: Buffer cinta/ desplazamiento pantalla
EAL	00AE-00AF	174-175	Direcc. fin cinta / fin de programa
CMP0	00B0-00B1	176-177	Constantes temporización cinta
TAPE1	00B2-00B3	178-179	Puntero: inicio de buffer cinta
BITTS	00B4	180	Contador bit fuera RS-232 / temp. cassette
NXTBIT	00B5	181	Siguiente bit a enviar RS-232 / Flag EOT cinta
RODATA	00B6	182	Buffer byte fuera RS-232
FNLEN	00B7	183	Longitud de nombre actual archivo
LA	00B8	184	Número actual archivo lógico



Con el fascículo anterior se han puesto a la venta las tapas correspondientes al octavo volumen



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 8, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:



1. Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.
2. Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.
3. Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.

Editorial Delta

Cada sobre de transferibles contiene una serie completa de números, del 1 al 8, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

Ya están a su disposición, en todos los quioscos y librerías, las tapas intercambiables para encuadernar 12 fascículos de

MI COMPUTER

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda

MI COMPUTER

Editorial Delta



10096

9 788485 822836